```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

B 16

```
**FILE**ID**RMOXPFN

RRRRRRRR    MM      MM    000000    XX      XX  PPPPPPPP    FFFFFFFFFF  NN      NN
RRRRRRRR    MM      MM    000000    XX      XX  PPPPPPPP    FFFFFFFFFF  NN      NN
RR      RR  MMMM  MMMM  00      00  XX      XX  PP      PP  FF          NN      NN
RR      RR  MMMM  MMMM  00      00  XX      XX  PP      PP  FF          NN      NN
RR      RR  MM  MM  MM  00    0000    XX  XX    PP      PP  FF          NNNN    NN
RR      RR  MM  MM  MM  00    0000    XX  XX    PP      PP  FF          NNNN    NN
RRRRRRRR    MM      MM  00  00  00      XX      PPPPPPPP    FFFFFFFF    NN  NN  NN
RRRRRRRR    MM      MM  00  00  00      XX      PPPPPPPP    FFFFFFFF    NN  NN  NN
RR  RR      MM      MM  0000    00    XX  XX    PP          FF          NN    NNNN
RR  RR      MM      MM  0000    00    XX  XX    PP          FF          NN    NNNN
RR    RR    MM      MM  00      00  XX      XX  PP          FF          NN      NN      ....
RR    RR    MM      MM  00      00  XX      XX  PP          FF          NN      NN      ....
RR      RR  MM      MM    000000    XX      XX  PP          FF          NN      NN      ....
RR      RR  MM      MM    000000    XX      XX  PP          FF          NN      NN      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
    (2)       281         DECLARATIONS
    (3)       328         RM$XPFN, Filename Expansion Control Routine
    (6)       606         SETNAM, Fill in NAM block fields
    (7)       661         UPDATE_SLB, Search List Update Routine
    (8)       711         EXPAND_NAME, Apply File Name Strings
    (10)      883         STUFF_NAME, Allocate and set up SLBHs
    (11)      965         PARSE_SPECS, Parse the filename specs
    (13)     1104         PARSE_STRING, Parse Individual Filename String
    (20)     1533         CHECK_NODE, Check Node Specification
    (28)     2078         PARSE_DIR, Parse a Directory String
    (32)     2418         TRANSLATE, Translate Logical Name
    (35)     2710         PARSE_CONCEAL, Parse Concealed Logical Name
    (37)     2813         ALLOCATE_SLB, Allocate Search List Block
    (38)     2893         UPCASE_COMPRESS Upcase and compress user strings
    (39)     3026         CHECK_FIELD - Check for Valid Field and Copy Routine
    (40)     3257         STICKY_DIR, Process Sticky Directories
```

```
0000            $BEGIN  RMOXPFN,000,RM$RMSFILENAME,<EXPAND FILENAME STRING>
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000
0000    ;++
0000    ;
0000    ; Facility: RMS
0000    ;
0000    ; Abstract:
0000    ;
0000    ;       This module expands the user supplied primary, default, and related
0000    ;       file name strings by the translation of logical names and the
0000    ;       application of defaults to form a fully qualified file specification.
0000    ;
0000    ; Environment: VAX/VMS, executive mode
0000    ;
0000    ; Author:       Keith B. Thompson,      Recreation Date: 7-Mar-1983
0000    ;
0000    ; Modified By:
0000    ;
0000    ;       V03-050 DGB0080         Donald G. Blair         21-Aug-1984
0000    ;               Fix bug in UPCASE_COMPRESS so that it doesn't write
0000    ;               past end of buffer when a quote is the last character
0000    ;               in a filename string.
0000    ;
0000    ;       V03-049 DGB0074         Donald G. Blair         02-Aug-1984
0000    ;               Fix node parsing so it remembers that the 2nd word
0000    ;               of a node descriptor may have flag bits set, i.e.
0000    ;               the length field is not a longword.
0000    ;
0000    ;       V03-048 DGB0073         Donald G. Blair         31-Jul-1984
0000    ;               Fix rooted directories so they are propagated correctly
0000    ;               from the related name block.
0000    ;
0000    ;       V03-047 RAS0321         Ron Schaefer            9-Jul-1984
```

```
0000     58 ;                   Eliminate access mode itemlist checking for PPF after
0000     59 ;                   translating logical names.   The check is not really
0000     60 ;                   sufficient and just takes needless CPU cycles.
0000     61 ;
0000     62 ;           V03-046 DGB0059         Donald G. Blair         27-Jun-1984
0000     63 ;                   Disallow concealed device names of the form
0000     64 ;                   "[dir-spec.]", i.e. those which are missing the
0000     65 ;                   device name.
0000     66 ;
0000     67 ;           V03-045 RAS0302         Ron Schaefer            30-Apr-1984
0000     68 ;                   Return RMS$_NOD if a node name logical translates to
0000     69 ;                   'node::xxxxx', this used to be accepted and 'xxxxx' was
0000     70 ;                   ignored.   Fix instruction datatype used.
0000     71 ;
0000     72 ;           V03-044 JEJ0029         J E Johnson             19-Apr-1984
0000     73 ;                   Make quoted network file specs check for other specification
0000     74 ;                   parts than filename, such as directory, device, type, or
0000     75 ;                   version.  This is illegal as the legal quoted spec is:
0000     76 ;                   node"access control string::"filename".
0000     77 ;
0000     78 ;           V03-043 JEJ0028         J E Johnson             11-Apr-1984
0000     79 ;                   Minor equate file cleanup.
0000     80 ;
0000     81 ;           V03-042 RAS0283         Ron Schaefer            28-Mar-1984
0000     82 ;                   Change the loop counter for RAS0266 to be 32767. (!)
0000     83 ;                   Only translate SYS$DISK if there in NOT a device name present
0000     84 ;                   already.  This is the same as V3 but causes the searchlist
0000     85 ;                   properties of SYS$DISK to be ignored if you have a device.
0000     86 ;                   Try to improve the performance of this beast a tad.
0000     87 ;                   Make explicit related-file rooted directories be ignored
0000     88 ;                   if duplicate.
0000     89 ;
0000     90 ;           V03-041 RAS0266         Ron Schaefer            12-Mar-1984
0000     91 ;                   Count the related NAM blocks that we parse. After processing
0000     92 ;                   255 of them, give up and return RMS$_RLF.  This is far more
0000     93 ;                   elegant than an infinite loop caused by invalid programs.
0000     94 ;
0000     95 ;           V03-040 RAS0261         Ron Schaefer            5-Mar-1984
0000     96 ;                   Fix a couple of small bugs and help performance in RM$XPFN.
0000     97 ;                     1. Make related file node,dev and dir only be ignored if an
0000     98 ;                        explicit nodename (null or otherwise) was seen.
0000     99 ;                     2. Make FWA$V_NAM_DVI stay set so that concealed DVI devices
0000    100 ;                        stay that way.
0000    101 ;                     3. Make sure related sticky "..." directoriesare copied.
0000    102 ;                     4. Remove quoted string internal parsing from this module
0000    103 ;                        and put it into the network access code.
0000    104 ;
0000    105 ;           V03-039 RAS0253         Ron Schaefer            15-Feb-1984
0000    106 ;                   Convert node logical name translation to use $TRNLNM.
0000    107 ;                   Node logicals can not contain searchlists.
0000    108 ;                   Convert the upcase algorithm to use EXE$UPCASE_DAT.
0000    109 ;                   Ignore related node, device and dir fields when a
0000    110 ;                   node or null node is present.
0000    111 ;                   Eliminate leading "_" manipulations, fix error return
0000    112 ;                   for node-name logical RMS$_LNE errors.
0000    113 ;
0000    114 ;           V03-038 RAS0258         Ron Schaefer            24-Feb-1984
```

```
0000  ;                      Fix RAS0256 and RAS0257 to only ignore related file PPFs.
0000  ;
0000  ;              V03-037 RAS0257          Ron Schaefer            23-Feb-1984
0000  ;                      Fix RAS0256 so that concealed devices work correctly.
0000  ;
0000  ;              V03-036 RAS0256          Ron Schaefer            21-Feb-1984
0000  ;                      Ignore PPFs that are from the DNA, RLFi or SET DEFAULT strings.
0000  ;
0000  ;              V03-035 RAS0248          Ron Schaefer            4-Feb-1984
0000  ;                      Ignore nul fields for input related filespecs.
0000  ;
0000  ;              V03-034 RAS0234          Ron Schaefer            11-Jan-1984
0000  ;                      Add support for SLBHs, revise filename string storage
0000  ;                      in order to implement multi-line input sticky searchlists.
0000  ;                      Revise searchlist processing rules to incorporate
0000  ;                      arbitrary many searchlist sources: FN, DN, RNi, and SD.
0000  ;                      Any compatibility of filename parsing between V3 and V4
0000  ;                      is purely coincidental after all this!
0000  ;
0000  ;              V03-033 RAS0238          Ron Schaefer            18-Jan-1984
0000  ;                      Fix bad R11 value for allocating space to store the
0000  ;                      filename strings.  (Fix to RAS0219).
0000  ;
0000  ;              V03-032 RAS0231          Ron Schaefer            9-Jan-1984
0000  ;                      Add support for NAM$V_SYNCHK (part 1); copy the NAM$V_SYNCHK
0000  ;                      bit into FWA$V_SYNTAX_CHK for later use both here
0000  ;                      and in $PARSE.
0000  ;
0000  ;              V03-031 RAS0219          Ron Schaefer            8-Dec-1983
0000  ;                      Revamp FWA structures for better memory use;
0000  ;                      make translation buffers be dynamic.
0000  ;
0000  ;              V03-030 RAS0208          Ron Schaefer            3-Nov-1983
0000  ;                      Make sticky directories work for UIC-format directories.
0000  ;                      The STICKY_DIR and MOVE_RELATED subroutines were
0000  ;                      incompletely integrated.
0000  ;
0000  ;              V03-029 RAS0203          Ron Schaefer            18-Oct-1983
0000  ;                      Fix network node parsing to not set the ACS flag
0000  ;                      in all cases, but only when it is actually present.
0000  ;
0000  ;              V03-028 RAS0202          Ron Schaefer            17-Oct-1983
0000  ;                      Clear translation attribute when processing the next
0000  ;                      element of the searchlist.
0000  ;
0000  ;              V03-027 RAS0191          Ron Schaefer            12-Sep-1983
0000  ;                      Fix bugcheck caused by bogus searchlist structures
0000  ;                      created when "node::searchlist_devnam:" is encountered.
0000  ;                      The correction involves eliminating a real obscure hack
0000  ;                      in network filespec processing that allowed user-mode
0000  ;                      process table device names to be translated.
0000  ;
0000  ;              V03-026 RAS0184          Ron Schaefer            2-Sep-1983
0000  ;                      Fix KBT0573 to be a NOP for now until the $TRNLOG is
0000  ;                      converted to $TRNLNM for logical node names.
0000  ;
0000  ;              V03-025 KBT0581          Keith B. Thompson       11-Aug-1983
```

```
0000   ;               Change size of file name on magtape and fix a (yes
0000   ;               there was one) search list bug!
0000   ;
0000   ;       V03-024 KBT0573          Keith B. Thompson       3-Aug-1983
0000   ;               Remove last ref. to FAB$B_DSBMSK
0000   ;
0000   ;       V03-023 KBT0555          Keith B. Thompson       20-Jul-1983
0000   ;               Probe the input strings
0000   ;
0000   ;       V03-022 KBT0551          Keith B. Thompson       23-Jun-1983
0000   ;               FWA$B_LNFLG changed it's name
0000   ;
0000   ;       V03-021 KBT0545          Keith B. Thompson       22-Jun-1983
0000   ;               Fix an out of phase symbol
0000   ;
0000   ;       V03-020 KBT0535          Keith B. Thompson       25-May-1983
0000   ;               Use new logical name services and turn on search list
0000   ;
0000   ;       V03-019 KBT0514          Keith B. Thompson       23-May-1983
0000   ;               Put this monster in a remote psect
0000   ;
0000   ;       V03-018 KBT0505          Keith B. Thompson       3-May-1983
0000   ;               Add search list support
0000   ;
0000   ;       V03-017 RAS0150          Ron Schaefer            29-Apr-1983
0000   ;               Fix incorrect recursion error return path for
0000   ;               parsing the default directory.
0000   ;
0000   ;       V03-016 RAS0149          Ron Schaefer            28-Apr-1983
0000   ;               Fix wrong register usage for 'node::[]' and
0000   ;               get correct address for '[a.b...]'.
0000   ;
0000   ;       V03-015 KBT0500          Keith B. Thompson       7-Mar-1983
0000   ;               Rewrite PARSE_STRING, CHECK_FIELD, STICKY_DIR, MOVE_RELATED,
0000   ;               remove NEXT_FIELD etc...
0000   ;
0000   ;       V03-014 TMK0001          Todd M. Katz            28-Feb-1983
0000   ;               Change a BLSS to a BLSSU within the routine MOVE_RELATED.
0000   ;               When the maximum size of a directory specification went above
0000   ;               a hexadecimal 7F, this branch was always being taken. As a
0000   ;               result, sticky directories broke.
0000   ;
0000   ;       V03-013 LJA0063          Laurie J. Anderson      23-Feb-1983
0000   ;               Move a couple routines involved with the NAM block to
0000   ;               RMONAMSTR.MAR where they belong.  They include RM$CHKNAMBLK
0000   ;               and RM$CHKNAM.
0000   ;
0000   ;       V03-012 KBT0493          Keith B. Thompson       11-Feb-1983
0000   ;               Add support for null node spec.
0000   ;
0000   ;       V03-011 KBT0443          Keith B. Thompson       3-Jan-1983
0000   ;               Fix support for "$" and "_" in directory names.
0000   ;
0000   ;       V03-010 SHZ0001          Stephen H. Zalewski,    5-Nov-1982  16:05
0000   ;               Support '$' and '_' in both filenames and filetypes.  Also
0000   ;               allow filenames and filetypes to be a max of 39 characters
0000   ;               each.
```

```
0000    ;
0000    ;       V03-009 KBT0219         Keith B. Thompson       23-Aug-1982
0000    ;               Reorganize psects
0000    ;
0000    ;       V03-008 KRM0057         K Malik                 10-Aug-1982
0000    ;               Change ASSUME statement in logical nodename translation
0000    ;               section to refer to new FWA$T_NODEBUF & FWA$B_UNDER_NOD
0000    ;               symbols.
0000    ;
0000    ;       V03-007 DMW4002         DMWalp                  1-Aug-1982
0000    ;               Correct problem where if input string > 63 alpha-num
0000    ;               characters without any punctuation then string was
0000    ;               dropped on the floor without an error because $TRNLOG
0000    ;               only copies the input to the output string on NOTRANS
0000    ;               error.
0000    ;
0000    ;       V03-006 KBT0100         Keith B. Thompson       13-Jul-1982
0000    ;               Clean up psects
0000    ;
0000    ;       V03-005 KRM0015         K R Malik               22-Apr-1982
0000    ;               For a network filespec, stop resolving leading hyphens in the
0000    ;               directory string (at the local node), because the hyphen has
0000    ;               meaning only in the context of the default directory at the
0000    ;               remote node. For example, in a directory string of the form
0000    ;               [-.A.B], the hyphen should be passed to FAL.
0000    ;
0000    ;       V03-004 JAK0076         J A Krycka              20-Apr-1982
0000    ;               Scan foreign filespec within a quoted string for wildcard
0000    ;               characters, and set the NAM$V_WILDCARD bit if any found.
0000    ;
0000    ;       V03-003 JAK0073         J A Krycka              12-Apr-1982
0000    ;               Eliminate the practice of prefixing a <del> character to a
0000    ;               resultant name string to indicate that the password has been
0000    ;               masked out of the nodespec. If the resultant name string is
0000    ;               then used as a related name string this flag character was
0000    ;               used to tell RMS to translate the nodespec string to obtain
0000    ;               the original nodespec with the real password. Now NXTFLD will
0000    ;               examine the access control string for a substring of 'pasword'
0000    ;               (setting V_PWD if found) to signal that the nodespec string
0000    ;               should be translated.
0000    ;
0000    ;       V03-002 KEK0029         K. E. Kinnear           24-Mar-1982
0000    ;               Return FNM error when we first get a quoted string (which
0000    ;               we think is ANSI) and THEN get a node spec later.  Always
0000    ;               used to give an FNM error before the advent of ANSI strings.
0000    ;
0000    ;       V03-001 RAS0079         Ron Schaefer            17-Mar-1982
0000    ;               Add support for translation table mask (FAB$B_DSBMSK).
0000    ;               All three $TRNLOG services are affected.
0000    ;
0000    ;--
```

```
                  0000           .SBTTL  DECLARATIONS
                  0000
                  0000   ;
                  0000   ; Include Files:
                  0000   ;
                  0000
                  0000           $FABDEF
                  0000           $FSCBDEF
                  0000           $FWADEF
                  0000           $IFBDEF
                  0000           $LNMDEF
                  0000           $LOGDEF
                  0000           $NAMDEF
                  0000           $NWADEF
                  0000           $RMSDEF
                  0000           $SLBDEF
                  0000           $SLBHDEF
                  0000           $SSDEF
                  0000           $TRNLOGDEF
                  0000
                  0000   ;
                  0000   ; Macros:
                  0000   ;
                  0000
                  0000   ;
                  0000   ; Equated Symbols:
                  0000   ;
                  0000
        00000020  0000           FOP             = FAB$L_FOP * 8                 ; Bit offset to FOP
        00000010  0000           WC              = FWA$B_WILDFLGS * 8            ; Bit offset to wild card fl
                  0000           WCNTV_MASK      = <1@<FWA$V_WC_NAME-WC>>!-
                  0000                             <1@<FWA$V_WC_TYPE-WC>>!-
        00000038  0000                             <1@<FWA$V_WC_VER-WC>>         ; Wild name, type,
                  0000                                                           ;  or ver mask
                  0000
                  0000   ;
                  0000   ; Other definitions:
                  0000   ;
                  0000
        0000001B  0000           ESCAPE          = ^X1B          ; ASCII escape character
        00000009  0000           HOR_TAB         = ^X09          ; ASCII value for horizontal tab
                  0000
                  0000   ;
                  0000   ; Own Storage:
                  0000   ;
                  0000
```

```
0000           .SBTTL  RM$XPFN, Filename Expansion Control Routine
0000
0000    ;++
0000    ;
0000    ; RM$XPFN - four major routines are included here:
0000    ;
0000    ;        1.  RM$XPFN     -        high-level control routine to select
0000    ;                                 and verify the various strings to
0000    ;                                 be used to expand the filename.
0000    ;        2.  PARSE_STRING -       medium-level routine to parse an
0000    ;                                 individual string into its
0000    ;                                 constituent parts, translating
0000    ;                                 logical names, and merging the
0000    ;                                 results into the expanded string.
0000    ;        3.  CHECK_FIELD -        low-level subroutine that
0000    ;                                 performs checks and other
0000    ;                                 functions as directed by PARSE_STRING.
0000    ;
0000    ; The high-level control routine, RM$XPFN performs the following functions:
0000    ;
0000    ;        1.  probes the filename string and calls PARSE_STRING to parse it.
0000    ;
0000    ;        2.  if not all filename fields (device, directory, file-name,
0000    ;            file-type, and file-version) are present applies defaults
0000    ;            in the order:
0000    ;
0000    ;                - default filename string (if any)
0000    ;                - related file (if any) for filename and/or filetype only
0000    ;                - default device by applying the logical name 'SYS$DISK:'
0000    ;                  (only if node not specified)
0000    ;                - default file directory as currently established
0000    ;                  (by login or the set default command) (only if
0000    ;                  node not specified)
0000    ;
0000    ;        3.  if a NAM block is specified and it includes an expanded
0000    ;            string buffer, the expanded string will be copied to
0000    ;            the buffer.
0000    ;
0000    ; Calling Sequence:
0000    ;
0000    ;        BSBW    RM$XPFN
0000    ;
0000    ; Input Parameters:
0000    ;
0000    ;        R8      FAB address
0000    ;        R9      IFAB address
0000    ;
0000    ; Implicit Inputs:
0000    ;
0000    ;        The contents of the FAB and related NAM block if any,
0000    ;        in particular the fields FNA, FNS, DNA, DNS, RLF, and NAM.
0000    ;                - if RLF specified, NAM in the related FAB and
0000    ;                  RSA and RSL in the NAM block
0000    ;                - if NAM, ESA and ESS
0000    ;
0000    ;        Also, the contents of the various logical name tables,
0000    ;        Especially with respect to the various system-defined names.
```

```
0000  ;
0000  ;         Also, the current default directory string.
0000  ;
0000  ;         FWA and NWA inputs:
0000  ;
0000  ;         FWA$Q_NODE      - descriptor of 127-char buffer in FWA
0000  ;         FWA$Q_DEVICE    - descriptor of  15-char buffer in FWA
0000  ;         FWA$Q_DIR1      - descriptor of   9-char buffer in FWA
0000  ;         FWA$Q_DIR2      - descriptor of   9-char buffer in FWA
0000  ;         FWA$Q_DIR2+8    - ''
0000  ;         FWA$Q_DIR2+16   - ''
0000  ;         FWA$Q_DIR2+24   - ''
0000  ;         FWA$Q_DIR2+32   - ''
0000  ;         FWA$Q_DIR2+40   - ''
0000  ;         FWA$Q_DIR2+48   - ''
0000  ;         FWA$Q_NAME      - descriptor of  10-char buffer in FWA
0000  ;         FWA$Q_TYPE      - descriptor of   4-char buffer in FWA
0000  ;         FWA$Q_VERSION   - descriptor of   6-char buffer in FWA
0000  ;         FWA$Q_QUOTED    - descriptor of 127-char buffer in FWA
0000  ;         FWA$Q_NODE1     - descriptor using shared node buffer in NWA
0000  ;         FWA$Q_NODE1+8   - ''
0000  ;         FWA$Q_NODE1+16  - ''
0000  ;         FWA$Q_NODE1+24  - ''
0000  ;         FWA$Q_NODE1+32  - ''
0000  ;         FWA$Q_NODE1+40  - ''
0000  ;         FWA$Q_NODE1+48  - ''
0000  ;         FWA$Q_NODE1+56  - ''
0000  ;
0000  ;         Notes:
0000  ;                 1.  the name, type, and version buffers must be
0000  ;                     adjacent and ascending in memory
0000  ;
0000  ;                 2.  all other FWA fields should be zero
0000  ;
0000  ; Output Parameters:
0000  ;
0000  ;         R0      Status code
0000  ;         R1-R7   Destroyed
0000  ;         R10     FWA address
0000  ;         AP      Destroyed
0000  ;
0000  ; Implicit Outputs:
0000  ;
0000  ;         NAM block (if specified):
0000  ;                 -if ESA specified buffer filled in with
0000  ;                  expanded filename string + ESL set to its
0000  ;                  length in bytes.
0000  ;
0000  ;         FWA:    -various flags set as per the parse
0000  ;                 -FWA$B_DIRTERM set to directory terminating bracket (']' or '>')
0000  ;                 -FWA$L_ESCSTRING set to escape string if seen
0000  ;                 -the descriptors for the parsed filename elements
0000  ;                  are set with the sizes of the various elements
0000  ;                  of the filename string, the elements themselves
0000  ;                  having been copied to the buffers. The filetype
0000  ;                  and version are appended to the file name.
0000  ;
```

```
                                 0000   442 ; Completion Codes:
                                 0000   443 ;
                                 0000   444 ;       Standard RMS completion codes, including:
                                 0000   445 ;               SUC, FNA, DNA, RLF, SYN, NOD, DEV, DIR, FNM, TYP, VER, LNE,
                                 0000   446 ;               QUO, NAM, ESA, ESS
                                 0000   447 ;
                                 0000   448 ; Side Effects:
                                 0000   449 ;
                                 0000   450 ;       None
                                 0000   451 ;
                                 0000   452 ;--
                                 0000   453
                                 0000   454         ASSUME  2*<LNM$C_NAMLENGTH+1> LE 512      ; 2 xlt buffers per page
                                 0000   455         ASSUME  NAM$C_MAXRSS LE 256               ; room for 1 copy of name
                                 0000   456
                                 0000   457 XPFN_SPACE = <  <2*<LNM$C_NAMLENGTH+1>>+ -        ; space for 2 xlt buffs
                                 0000   458                 <2*<FSCB$K_BLN>>+ -               ; space for 2 FSCBs
                 00000508        0000   459                 256  >                            ; space for name string
                                 0000   460
                                 0000   461 RM$XPFN::
                                 0000   462         $TSTPT  XPFN
           00000000'EF     16    0006   463         JSB     RM$FWASET                         ; get a FWA
                 48 50     E9    000C   464         BLBC    R0,10$                            ; branch on failure
                                 000F   465
          52   0508 8F     3C    000F   466         MOVZWL  #XPFN_SPACE,R2                    ; size of all the scratch space
           00000000'EF     16    0014   467         JSB     RM$GETPAG                         ; get scratch space
                 3A 50     E9    001A   468         BLBC    R0,10$                            ; room?
                                 001D   469
          00A0 CA   63     9E    001D   470         MOVAB   (R3),FWA$L_XLTBUFF1(R10)          ; 1st buffer
     00A4 CA     0100 C3   9E    0022   471         MOVAB   <1+LNM$C_NAMLENGTH>(R3),FWA$L_XLTBUFF2(R10) ; 2nd buffer
                 0200 C3   9E    0029   472         MOVAB   <2*<LNM$C_NAMLENGTH+1>>(R3),-     ; space for 2 xlt buffs
                   50 AA         002D   473                 FWA$L_BUF_PTR(R10)                ; save scratch buffer address
                                 002F   474
             54 AA  5B     D0    002F   475         MOVL    R11,FWA$L_IMPURE_AREA(R10)        ; save impure address
                 0300 C3   9E    0033   476         MOVAB   <256+<2*<[NM$C_NAMLENGTH+1>>>(R3),-
                      5B         0037   477                 R11                               ; point to 1st FSCB
                   0141    30    0038   478         BSBW    EXPAND_NAME                       ; do bulk of work
            5B    54 AA    D0    003B   479         MOVL    FWA$L_IMPURE_AREA(R10),R11        ; restore impure address
                      50   DD    003F   480         PUSHL   R0                                ; save status
                                 0041   481
          55   0508 8F     3C    0041   482         MOVZWL  #XPFN_SPACE,R5                    ; size of all the scratch space
          54   00A0 CA     D0    0046   483         MOVL    FWA$L_XLTBUFF1(R10),R4            ; addr of space
           00000000'EF     16    004B   484         JSB     RM$RETPAG                         ; return it all
                   50 8ED0       0051   485         POPL    R0                                ; restore status
                 01 50     E8    0054   486         BLBS    R0,20$
                           05    0057   487 10$:    RSB                                       ; exit with error
```

```
                                 0058    489 ;
                                 0058    490 ; Expand_name was successful!
                                 0058    491 ;
                                 0058    492 ; Check to see if there the first user directory is the MFD and flags it
                                 0058    493 ; as such.  NOTE: We define '000000' as being a reserve word meaning
                                 0058    494 ; the MFD which can be the real '000000.DIR' or the rooted directory.
                                 0058    495 ;
                                 0058    496
      56    0130 CA      7D      0058    497 20$:        MOVQ    FWA$Q_DIR1(R10),R6                ; get first directory name
               56 06     B1      005D    498             CMPW    #6,R6                             ; is it the correct size
                  16     12      0060    499             BNEQ    40$                               ; can't be if not 6
 87      30303030 8F     D1      0062    500             CMPL    #^A/0000/,(R7)+                   ; is it the form of
                  0D     12      0069    501             BNEQ    40$                               ;  '000000'
      67    3030 8F      B1      006B    502             CMPW    #^A/00/,(R7)                      ; check the last two bytes
                  06     12      0070    503             BNEQ    40$                               ; no continue
                                 0072    504             SSB     #FSCB$V_MFD,FWA$Q_DIR1(R10)       ; we have one, set the flag
                                 0078    505
                                 0078    506 ;
                                 0078    507 ; We have all the parts described separately.
                                 0078    508 ; Append file type and version to the filename.
                                 0078    509 ;
                                 0078    510
        28 6A     1A     E1      0078    511 40$:        BBC     #FWA$V_QUOTED,(R10),50$           ; check if network quoted
        0F 6A     19     E1      007C    512             BBC     #FWA$V_NODE,(R10),44$             ;  string found
                                 0080    513
                                 0080    514 ;
                                 0080    515 ; Network quoted string -- check for only filename and node specified.
                                 0080    516 ; note that version and type were checked during the $filescan.
                                 0080    517 ;
                                 0080    518
 6A      00C3C000 8F     D3      0080    519             BITL    #FWA$M_EXP_TYPE!FWA$M_EXP_VER!-
                                 0087    520                      FWA$M_EXP_DIR!FWA$M_EXP_DEV!-
                                 0087    521                      FWA$M_DIR!FWA$M_DEVICE,(R10)     ; test for all other spec parts
                  42     13      0087    522             BEQL    70$                               ; Branch if a legal spec
                                 0089    523             RMSERR  QUO
                         05      008E    524             RSB                                       ; Return with an error.
                                 008F    525
                                 008F    526 ;
                                 008F    527 ; Upcase quoted string in place -- can't use UPCASE_COMPRESS because:
                                 008F    528 ; we don't want to compress the string
                                 008F    529 ;
                                 008F    530
      50    0170 CA      7D      008F    531 44$:        MOVQ    FWA$Q_NAME(R10),R0                ; set up descriptors in R0,R1
                  0B     11      0094    532             BRB     46$                               ; enter loop in progress
                                 0096    533
            52    61     9A      0096    534 45$:        MOVZBL  (R1),R2                           ; get char
 81     00000000'GF42    90      0099    535             MOVB    G^EXE$UPCASE_DAT[R2],-
                                 00A1    536                     (R1)+                             ; Upcase it
               F2 50     F4      00A1    537 46$:        SOBGEQ  R0,45$                            ; count down characters
                                 00A4    538
      56    0170 CA      7E      00A4    539 50$:        MOVAQ   FWA$Q_NAME(R10),R6                ; get name descriptor
      53    86    86     C1      00A9    540             ADDL3   (R6)+,(R6)+,R3                    ; get address past name
                                 00AD    541
                                 00AD    542             ASSUME  FWA$Q_TYPE      EQ      <FWA$Q_NAME+8>
                                 00AD    543             ASSUME  FWA$Q_VERSION   EQ      <FWA$Q_TYPE+8>
                                 00AD    544 ;           ASSUME  FWA$T_TYPEBUF   EQ      <FWA$T_NAMEBUF+FWA$S_NAMEBUF>
                                 00AD    545 ;           ASSUME  FWA$T_VERBUF    EQ      <FWA$T_TYPEBUF+FWA$S_TYPEBUF>
```

```
                          00AD   546
          83   2E   90    00AD   547            MOVB    #^A/./,(R3)+                    ; append '.' to delimit type
               23   10    00B0   548            BSBB    MOVNXT                          ; append type
       08 68   3D   E1    00B2   549            BBC     #FAB$V_OFP+FOP,(R8),60$         ; branch if not output file parse
       04 6A   13   E1    00B6   550            BBC     #FWA$V_WC_VER,(R10),60$         ; branch if not wildcard version
        0180 CA    B7     00BA   551            DECW    FWA$Q_VERSION(R10)              ; get rid of '*' version
          83   3B   90    00BE   552    60$:    MOVB    #^A/;/,(R3)+                    ; append ';'
               12   10    00C1   553            BSBB    MOVNXT                          ; and version
0170 CA    53  0174 CA C3 00C3   554            SUBL3   FWA$Q_NAME+4(R10),R3,-          ; adjust length
                          00CB   555                    FWA$Q_NAME(R10)
                          00CB   556
                          00CB   557    ;
                          00CB   558    ; Modify node descriptors to include leading underscore where
                          00CB   559    ; Appropriate. Note that this code is skipped for process permanent files.
                          00CB   560    ;
                          00CB   561
                          00CB   562    70$:;;; bbc     #fwa$v_node,(r10),80$           ; branch if node name not seen
                          00CB   563    ;;;     incw    fwa$q_node(r10)                 ; modify node descriptor to
                          00CB   564    ;;;     decl    fwa$q_node+4(r10)               ;  include leading underscore
                          00CB   565    ;;;     incw    fwa$q_node1(r10)                ; modify node1 descriptor
                          00CB   566    ;;;     decl    fwa$q_node1+4(r10)
                          00CB   567    ;;;80$: bbc     #fwa$v_dev_under,-              ; branch if device name was not
                          00CB   568    ;;;             (r10),90$                       ;  prefixed by an underscore
                          00CB   569    ;;;     incw    fwa$q_device(r10)               ; modify device descriptor to
                          00CB   570    ;;;     decl    fwa$q_device+4(r10)             ;  include leading underscore
                          00CB   571
                          00CB   572    ;
                          00CB   573    ; If user has specified a NAM block and an expanded string buffer within it,
                          00CB   574    ; give him the expanded name.
                          00CB   575    ;
                          00CB   576
                          00CB   577    90$:    RMSSUC                                  ; declare success so far
       57   28 A8   D0    00CE   578            MOVL    FAB$L_NAM(R8),R7                ; is there a NAM block
               0F   12    00D2   579            BNEQ    SETNAM                          ; branch if yes
                    05    00D4   580            RSB                                     ; exit success if not
```

```
                          00D5  ;+
                          00D5  ;
                          00D5  ; Subroutine to append file type and version to the file name.
                          00D5  ;
                          00D5  ; Inputs:
                          00D5  ;
                          00D5  ;         R3      Destination address
                          00D5  ;         R6      Descriptor of field to append
                          00D5  ;
                          00D5  ; Outputs:
                          00D5  ;
                          00D5  ;         R0-R5   Destroyed
                          00D5  ;         R3      Address past last byte moved
                          00D5  ;         R6      = R6 + 8 (The descriptor referenced by R6 is updated to point
                          00D5  ;                         point to the moved string.)
                          00D5  ;
                          00D5  ;-
                          00D5
        50    86    D0    00D5  MOVNXT: MOVL    (R6)+,R0                        ; get length
        51    66    D0    00D8          MOVL    (R6),R1                         ; get address of string
        86    53    D0    00DB          MOVL    R3,(R6)+                        ; save destination address in desc
  63    61    50    28    00DE          MOVC3   R0,(R1),(R3)                    ; move field
                    05    00E2          RSB
```

```
                           00E3   606          .SBTTL  SETNAM, Fill in NAM block fields
                           00E3   607 ;
                           00E3   608 ; Fill in the expanded filename string and FNB status bits.
                           00E3   609 ;
                           00E3   610
            0139    30     00E3   611 SETNAM: BSBW    CHKNAMBLK                       ; check access to name block
       0F 68   38    E1    00E6   612         BBC     #FAB$V_NAM+FOP,(R8),15$         ; branch if not doing open by NAM block
          2A A7    D5      00EA   613         TSTL    NAM$W_DID(R7)                   ; was directory ID an input?
             56    12      00ED   614         BNEQ    XITNAM                          ; branch if yes (don't output anything)
          24 A7    D5      00EF   615         TSTL    NAM$W_FID(R7)                   ; do we have a file ID?
             51    12      00F2   616         BNEQ    XITNAM                          ; branch if yes
             11    E0      00F4   617         BBS     #NAM$V_NODE,-                   ; branch if network access
       4C 34 A7            00F6   618                 NAM$L_FNB(R7),XITNAM
                           00F9   619
                           00F9   620 ;
                           00F9   621 ; Return expanded name string to user's buffer.
                           00F9   622 ;
                           00F9   623
       5C    46'AF    9E   00F9   624 15$:    MOVAB   B^EXPARGL,AP                    ; Address of RM$EXPSTRING argument list
     00000000'EF      16   00FD   625         JSB     RM$EXPSTRING                    ; Return expanded name string
          3F 50       E9   0103   626         BLBC    R0,XITNAM                       ; Quit on error
                           0106   627
                           0106   628 ;
                           0106   629 ; Fill in the filename status bits.
                           0106   630 ;
                           0106   631
          34 A7    D4      0106   632 20$:    CLRL    NAM$L_FNB(R7)                   ; Clear the FNB
          02 AA    F0      0109   633         INSV    FWA$B_WILDFLGS(R10),-           ; Set low 9 bits
  34 A7    09  00          010C   634                 #0,#9,NAM$L_FNB(R7)             ;
          0C AA    95      0110   635         TSTB    FWA$B_ESCFLG(R10)               ; PPF flag set?
             05    13      0113   636         BEQL    30$                             ; Branch if not
                           0115   637         SSB     #NAM$V_PPF,NAM$L_FNB(R7); Yes, set flag
51    6A   07  19    EF    011A   638 30$:    EXTZV   #FWA$V_NODE,#7,(R10),R1         ; Get next 7 bits
      07   11  51    F0    011F   639         INSV    R1,#NAM$V_NODE,#7,-             ;  and set them
          34 A7            0123   640                 NAM$L_FNB(R7)                   ;
                           0125   641
                           0125   642         ASSUME  NAM$V_WILD_UFD  EQ      24
                           0125   643
          05 AA    90      0125   644         MOVB    FWA$B_DIRWCFLGS(R10),-          ; Set directory wildcard flags
          37 A7            0128   645                 NAM$L_FNB+3(R7)                 ;
       05 6A   38    E1    012A   646         BBC     #FWA$V_SLPRESENT,(R10),40$          ; search list present?
                           012E   647         SSB     #NAM$V_SEARCH_LIST,NAM$L_FNB(R7); flag it in name block
       05 6A   39    E1    0133   648 40$:    BBC     #FWA$V_CONCEAL_DEV,(R10),50$        ; concealed device present?
                           0137   649         SSB     #NAM$V_CNCL_DEV,NAM$L_FNB(R7)       ;  flag it in name block
       05 6A   3A    E1    013C   650 50$:    BBC     #FWA$V_ROOT_DIR,(R10),XITNAM        ; root directory present?
                           0140   651         SSB     #NAM$V_ROOT_DIR,NAM$L_FNB(R7)       ;  flag it in name block
                           0145   652
                   05      0145   653 XITNAM: RSB                                     ; Return to caller of RM$XPFN
                           0146   654
                           0146   655 EXPARGL:                                        ;
                   0C      0146   656         .BYTE   NAM$L_ESA                       ; Argument list for RM$EXPSTRING
                           0147   657         RMSERR_WORD     ESA                     ;
                           0149   658         RMSERR_WORD     ESS                     ;
                           014B   659
```

```
                        014B   661          .SBTTL  UPDATE_SLB, Search List Update Routine
                        014B   662
                        014B   663 ;++
                        014B   664 ;
                        014B   665 ; UPDATE_SLB -
                        014B   666 ;
                        014B   667 ;        This routine updates the last search list block (SLB) in the
                        014B   668 ;        specified SLB list.  If there are no SLBs return failure.
                        014B   669 ;        If the last SLB cannot be updated it removes the SLB and tries
                        014B   670 ;        to update a previous SLB until all SLB are exausted.
                        014B   671 ;
                        014B   672 ; Input:
                        014B   673 ;
                        014B   674 ;        R6 - pointer to the SLBH list to update
                        014B   675 ;
                        014B   676 ; Output:
                        014B   677 ;
                        014B   678 ;        z-bit : clear - update success
                        014B   679 ;                set   - update failed
                        014B   680 ;
                        014B   681 ;--
                        014B   682
                        014B   683 UPDATE_SLB:
   54    0C A6    D0    014B   684          MOVL    SLBH$L_SLB_QUE(R6),R4             ; get first SLB
            2A    13    014F   685          BEQL    30$                               ; there is none (z-bit set)
   54    04 A4    D0    0151   686          MOVL    SLB$L_BLINK(R4),R4                ; get last SLB in list
         0C A4    D1    0155   687          CMPL    SLB$L_INDEX(R4),-                 ; have we gone past last
         10 A4          0158   688                  SLB$L_MAX_INDEX(R4)               ;  name in the search list
            19    19    015A   689          BLSS    20$                               ; no
                        015C   690
                        015C   691 ;
                        015C   692 ; The SLB we updated is no longer usefull, need to delete it and try to
                        015C   693 ; update the previous one
                        015C   694 ;
                        015C   695
      54    64    0F    015C   696          REMQUE  (R4),R4                           ; remove this SLB
            03    1C    015F   697          BVC     10$                               ; was que empty
         0C A6    D4    0161   698          CLRL    SLBH$L_SLB_QUE(R6)                ; yes, clear the list pointer
            5B    DD    0164   699 10$:     PUSHL   R11                               ; save FSCB
   5B    54 AA    D0    0166   700          MOVL    FWA$L_IMPURE_AREA(R10),R11        ; restore impure address
  00000000'EF     16    016A   701          JSB     RM$RETBLK1                        ; deallocate the space
            5B  8ED0    0170   702          POPL    R11                               ; restore FSCB
            D6    11    0173   703          BRB     UPDATE_SLB                        ; try again
                        0175   704
         14 A4    D4    0175   705 20$:     CLRL    SLB$L_ATTR(R4)                    ; clear attributes for next element
         0C A4    D6    0178   706          INCL    SLB$L_INDEX(R4)                   ; update index for next element
                        017B   707                                                    ;  (z-bit clear)
                  05    017B   708 30$:     RSB                                       ; return
                        017C   709
```

```
                     017C                  .SBTTL  EXPAND_NAME, Apply File Name Strings
                     017C
                     017C        ;++
                     017C        ;
                     017C        ; EXPAND_NAME - controls the parsing of the primary, default, and related
                     017C        ;        filename strings and the application of the various defaults.
                     017C        ;
                     017C        ;        It performs these functions by selecting the next string for parsing,
                     017C        ;        verifying its readability, and calling PARSE_STRING to include it.
                     017C        ;
                     017C        ;        Same inputs as for RM$XPFN.
                     017C        ;
                     017C        ;        Same outputs except that the expanded name string is not built up from
                     017C        ;        the parts.
                     017C        ;
                     017C        ;--
                     017C
                     017C        EXPAND_NAME:
    1C 6A    02  E1  017C                  BBC     #FWA$V_SL_PASS,(R10),10$        ; branch if this is not
                     0180                                                          ;  a search list pass
                     0180
                     0180        ;
                     0180        ; If this is a search list operation then try to update the search list
                     0180        ; blocks in the following order: FN, DN, RNi and SD.  If no update succedes
                     0180        ; then return error - 'no more strings'
                     0180        ;
                     0180
                     0180                  ASSUME  SLBH$L_FLINK     EQ      0
                     0180
57   00B0 CA     DE  0180                  MOVAL   FWA$L_SLBH_FLINK(R10),R7; start of que addr for end test
 56    67        D0  0185                  MOVL    (R7),R6                 ; start at file name list head
 57    56        D1  0188        1$:       CMPL    R6,R7                   ; back to beginning?
       09        13  018B                  BEQL    2$                      ; no search lists left
       BC        10  018D                  BSBB    UPDATE_SLB              ; update or delete this list
       08        12  018F                  BNEQ    5$                      ; if done, procede
 56    66        D0  0191                  MOVL    (R6),R6                 ; advance to next
       F2        11  0194                  BRB     1$
                     0196
       50        D4  0196        2$:       CLRL    R0                      ; return error 'no more list'
                 05  0198                  RSB                             ; exit
                     0199
     0182        31  0199        5$:       BRW     PARSE_SPECS             ; continue by parsing the strings
                     019C
                     019C                  ASSUME  FWA$L_SLBH_PTR+4 EQ FWA$L_SLB_PTR
                     019C
     00A8 CA     7C  019C        10$:      CLRQ    FWA$L_SLBH_PTR(R10)     ; clear the 'current' SLB[H] ptrs
                     01A0
                     01A0        ;
                     01A0        ; Check for open by name block
                     01A0        ;
                     01A0
57     28 A8     D0  01A0                  MOVL    FAB$L_NAM(R8),R7        ; get NAM block address
       4B        13  01A4                  BEQL    40$                     ; don't have one, need to parse normally
       77        10  01A6                  BSBB    CHKNAMBLK               ; check out the name block
       03        E1  01A8                  BBC     #NAM$V_SYNCHK,-         ; remember the syntax-only check flag
  04 08 A7           01AA                          NAM$B_NOP(R7),20$       ; for more convenient questioning
                     01AD                  SSB     #FWA$V_SYNTAX_CHK,(R10) ; set corresponding FWA bit
```

```
                        01B1
                        01B1  ;
                        01B1  ; It's a open by name block and there is a name block, so check for non-zero
                        01B1  ; device ID in the name block and if found use it to provide a device name.
                        01B1  ;
                        01B1
      30 68    38   E1  01B1  20$:    BBC     #FAB$V_NAM+FOP,(R8),30$ ; branch if not open by name block
      55    14 A7   9E  01B5          MOVAB   NAM$T_DVI(R7),R5        ; get device ID string address
         56    85   9A  01B9          MOVZBL  (R5)+,R6                ; get string length
               36   13  01BC          BEQL    50$                     ; branch if zero
         0F    56   D1  01BE          CMPL    R6,#15                  ; device ID > 15 chars?
               69   1A  01C1          BGTRU   ERRDVI                  ; branch if so
      57    50 AA   D0  01C3          MOVL    FWA$L_BUF_PTR(R10),R7   ; get address of scratch buffer for DVI
  67     65    56   28  01C7          MOVC3   R6,(R5),(R7)            ; move the string
         63    3A   90  01CB          MOVB    #^A/:/,(R3)             ; append colon to device name
               56   D6  01CE          INCL    R6                      ;  and count it
             0200   30  01D0          BSBW    PARSE_STRING            ; parse device name string
         6A    20   88  01D3          BISB2   #FWA$M_NAM_DVI,(R10)    ; flag device came from NAM block
      57    28 A8   D0  01D6          MOVL    FAB$L_NAM(R8),R7        ; get NAM block address again
               43   10  01DA          BSBB    CHKNAMBLK               ; check out the name block
            24 A7   D5  01DC          TSTL    NAM$W_FID(R7)           ; is file ID non-zero?
               10   13  01DF          BEQL    40$                     ; continue parse if not
      37 69    32   E1  01E1          BBC     #IFB$V_CREATE,(R9),60$  ; all done unless create
            24 A7   D4  01E5  30$:    CLRL    NAM$W_FID(R7)           ; zero the file ID for create
            0B A7   94  01E8          CLRB    NAM$B_ESL(R7)           ; zero returned strings
            03 A7   94  01EB          CLRB    NAM$B_RSL(R7)           ;
            34 A7   D4  01EE          CLRL    NAM$L_FNB(R7)           ; and status bits in case of error
             0082   31  01F1  40$:    BRW     STUFF_NAME              ; continue parse
                        01F4
                        01F4  ;
                        01F4  ; Don't allow DID or FID to be used since DVI is bad.
                        01F4  ;
                        01F4
            2A A7   D4  01F4  50$:    CLRL    NAM$W_DID(R7)           ; zero the directory ID
               11   E1  01F7          BBC     #NAM$V_NODE,-           ; branch if not network access
         E9 34 A7       01F9                  NAM$L_FNB(R7),30$
            24 A7   D4  01FC          CLRL    NAM$W_FID(R7)           ; zero the file ID
                        01FF
                        01FF  ;
                        01FF  ; Open by Name Block is not supported over the network, but it can be simulated
                        01FF  ; by using the resultant name string returned from a previous $SEARCH, $OPEN,
                        01FF  ; etc. This facilitates network wildcard support as many utilities open by
                        01FF  ; Name Block after finding the next file via a $SEARCH operation.
                        01FF  ;
                        01FF
      56    03 A7   9A  01FF          MOVZBL  NAM$B_RSL(R7),R6        ; get resultant string length
               E0   13  0203          BEQL    30$                     ;  and branch if zero (i.e., ignore)
      57    04 A7   D0  0205          MOVL    NAM$L_RSA(R7),R7        ; get resultant string address
               DA   13  0209          BEQL    30$                     ;  and branch if zero (i.e., ignore)
 67   56    0A A9   0C  020B          PROBER  IFB$B_MODE(R9),R6,(R7)  ; probe readability
               20   13  0210          BEQL    ERRRST                  ; branch if inaccessible
      55    50 AA   D0  0212          MOVL    FWA$L_BUF_PTR(R10),R5   ; get address of scratch buffer
             0AFC   30  0216          BSBW    UPCASE_COMPRESS         ; perform upcasing and string
                        0219                                          ;  compression while copying string
             01B7   30  0219          BSBW    PARSE_STRING            ; parse the string
             01A4   31  021C  60$:    BRW     ENDPRS                  ; all done even if all file name
                        021F                                          ;  parts are not present
```

```
                      021F  ;
                      021F  ; Check the nam block, if there is an error pop the pc and return
                      021F  ;
                      021F
                      021F  CHKNAMBLK:
      00000000'EF  16 021F          JSB     RM$CHKNAM                               ; check the name block
            03 50  E8 0225          BLBS    R0,10$                                  ; was there an error
            51 8ED0   0228          POPL    R1                                      ; yes pop return pc
                   05 022B  10$:    RSB                                             ; exit
                      022C
                      022C  ;
                      022C  ; Handle errors
                      022C  ;
                      022C
                      022C  ERRDVI: RMSERR  DVI                             ; Invalid device ID field
                   05 0231          RSB
                      0232
                      0232  ERRRST: RMSERR  RST                             ; Can't access resultant name
                   05 0237          RSB
                      0238
                      0238  ERRFNA: RMSERR  FNA                             ; Can't access filename string
                   05 023D          RSB
                      023E
                      023E  ERRDNA: RMSERR  DNA                             ; Can't access default filename string
                   05 0243          RSB
                      0244
                      0244  ; subroutine to get space for storing filename input strings,
                      0244  ; upcase/compress the string into the buffer and return updated size.
                      0244  ;
                      0244  ; inputs:
                      0244  ;               R6 - input size
                      0244  ;               R7 - input addr
                      0244  ; outputs:
                      0244  ;               R0 - status
                      0244  ;               R1 - ptr to space
                      0244  ;               R6 - output size
                      0244  ;               R7 - output addr
                      0244  ;
                      0244  ;               R2, R3, R4, R5 - scratch
                      0244  ;
                      0244
                      0244  GETSPC:
   52    56   14   C1 0244          ADDL3   #SLBH$C_BLN,R6,R2                       ; get space to store it
              5B   DD 0248          PUSHL   R11                                     ; save FSCB
     5B   54 AA    D0 024A          MOVL    FWA$L_IMPURE_AREA(R10),R11              ; restore impure address
      00000000'EF  16 024E          JSB     RM$GETSPC1                              ;
            5B 8ED0   0254          POPL    R11                                     ; restore FSCB
          1B 50    E9 0257          BLBC    R0,10$
      08 A1   2B   90 025A          MOVB    #SLBH$C_BID,SLBH$B_BID(R1)              ; identify structure
      09 A1   56   90 025E          MOVB    R6,SLBH$B_BLN(R1)                       ; size of string allocation
   55    14 A1     DE 0262          MOVAL   SLBH$T_STRING(R1),R5                    ; buffer data ptr
           0AAC    30 0266          BSBW    UPCASE_COMPRESS                         ; copy the name
      0B A1   56   90 0269          MOVB    R6,SLBH$B_STR_LEN(R1)                   ; stuff the size of string
   00B4 DA    61   0E 026D          INSQUE  (R1),@FWA$L_SLBH_BLINK(R10)             ; insert at tail of list
                      0272          RMSSUC
                   05 0275  10$:    RSB
```

```
                             0276   883         .SBTTL  STUFF_NAME, Allocate and set up SLBHs
                             0276   884 ;
                             0276   885 ; Stuff the fwa buffers with the file name, default name and related name
                             0276   886 ; strings from the FAB and related name blocks if any
                             0276   887 ;
                             0276   888
                             0276   889 STUFF_NAME:
                             0276   890
                             0276   891 ;
                             0276   892 ; Get the FNA string
                             0276   893 ;
                             0276   894
           56    34 A8    9A 0276   895         MOVZBL  FAB$B_FNS(R8),R6                ; get size of input string
                    16    13 027A   896         BEQL    10$                             ; skip if zero
           57    2C A8    D0 027C   897         MOVL    FAB$L_FNA(R8),R7                ; get input addr
                    10    13 0280   898         BEQL    10$                             ; skip if zero
     67    56    0A A9    0C 0282   899         PROBER  IFB$B_MODE(R9),R6,(R7)          ; prober buffer
                       AF 13 0287   900         BEQL    ERRFNA                          ; bad
                       B9 10 0289   901         BSBB    GETSPC                          ; get space
                    72 50 E9 028B   902         BLBC    R0,60$                          ; if room
        0A A1       10    90 028E   903         MOVB    #FWA$M_FNA_PASS,SLBH$B_PASSFLGS(R1) ; FWA$B_PASSFLGS value
                             0292   904
                             0292   905 ;
                             0292   906 ; Get the DNA string
                             0292   907 ;
                             0292   908
           56    35 A8    9A 0292   909 10$:    MOVZBL  FAB$B_DNS(R8),R6                ; get size of input string
                    16    13 0296   910         BEQL    20$                             ; skip if zero
           57    30 A8    D0 0298   911         MOVL    FAB$L_DNA(R8),R7                ; get input addr
                    10    13 029C   912         BEQL    20$                             ; skip if zero
     67    56    0A A9    0C 029E   913         PROBER  IFB$B_MODE(R9),R6,(R7)          ; prober buffer
                       99 13 02A3   914         BEQL    ERRDNA                          ; bad
                       9D 10 02A5   915         BSBB    GETSPC                          ; get space
                    56 50 E9 02A7   916         BLBC    R0,60$                          ; if room
        0A A1       01    90 02AA   917         MOVB    #FWA$M_DUPOK,SLBH$B_PASSFLGS(R1) ; FWA$B_PASSFLGS value
                             02AE   918
                             02AE   919 ;
                             02AE   920 ; Get the RLF strings
                             02AE   921 ;
                             02AE   922
           57    28 A8    D0 02AE   923 20$:    MOVL    FAB$L_NAM(R8),R7                ; get name block
                       59 13 02B2   924         BEQL    110$                            ; skip if none
                     FF68 30 02B4   925         BSBW    CHKNAMBLK                       ; check it (no return on error)
         7E    7FFF 8F    3C 02B7   926         MOVZWL  #32767,-(SP)                    ; RLF counter
           57    10 A7    D0 02BC   927 30$:    MOVL    NAM$L_RLF(R7),R7                ; get related nam block
                       48 13 02C0   928         BEQL    100$                            ; none
             00000000'EF  16 02C2   929         JSB     RM$CHKNAM                       ; check it
                    2D 50 E9 02C8   930         BLBC    R0,40$                          ; bad
           56    03 A7    9A 02CB   931         MOVZBL  NAM$B_RSL(R7),R6                ; get size of input string
                       39 13 02CF   932         BEQL    100$                            ; skip if zero
           55    04 A7    D0 02D1   933         MOVL    NAM$L_RSA(R7),R5                ; get input addr
                       33 13 02D5   934         BEQL    100$                            ; skip if zero
     65    56    0A A9    0C 02D7   935         PROBER  IFB$B_MODE(R9),R6,(R5)          ; prober buffer
                       1A 13 02DC   936         BEQL    40$                             ; bad
                       57 DD 02DE   937         PUSHL   R7                              ; save NAM blk addr
                 57    55 D0 02E0   938         MOVL    R5,R7                           ; put string in right reg
                     FF5E 30 02E3   939         BSBW    GETSPC                          ; get space
```

```
                  57 8ED0  02E6            POPL    R7                              ; retrieve NAM blk addr
               11 50   E9  02E9            BLBC    R0,50$                          ; if room
               09      90  02EC            MOVB    #<FWA$M_DUPOK!FWA$M_RLF_PASS>,-
            0A A1          02EE                    SLBH$B_PASSFLGS(R1)             ; FWA$B_PASSFLGS value
      10 A1    34 A7   D0  02F0            MOVL    NAM$L_FNB(R7),SLBH$L_NAM_FNB(R1); save FNB bits
               C4 6E   F5  02F5            SOBGTR  (SP),30$                        ; get next RLF
                           02F8
                           02F8    40$:    RMSERR  RLF                             ; Can't access related file name
                  51 8ED0  02FD    50$:    POPL    R1                              ; Discard RLF counter
                       05  0300    60$:    RSB
                           0301
                           0301    ;
                           0301    ; Setup the $SET DEFAULT string
                           0301    ;
                           0301
   3A 4B 53 49 44 24 53 59 53  0301    70$:    .ASCII  /SYS$DISK:/                     ; Default device string
                 00000009  030A    SYSDISKLEN = .-70$
                           030A
                  51 8ED0  030A    100$:   POPL    R1                              ; Discard RLF counter
          57    F1 AF  9E  030D    110$:   MOVAB   B^70$,R7                        ; addr of string
            56     09  9A  0311            MOVZBL  #SYSDISKLEN,R6                  ; size of string
                 FF2D  30  0314            BSBW    GETSPC                          ; set up SLBH for it
               E6 50   E9  0317            BLBC    R0,60$                          ; quit on error
         0A A1     01  90  031A            MOVB    #FWA$M_DUPOK,SLBH$B_PASSFLGS(R1) ; FWA$B_PASSFLGS value
```

```
                          031E          .SBTTL  PARSE_SPECS, Parse the filename specs
                          031E  ;
                          031E  ; Process the filename specifications in the order:
                          031E  ;         FNA,  DNA,  RLFi, SETDEFAULT
                          031E  ;
                          031E
                          031E  PARSE_SPECS:
                          031E
                          031E          ASSUME  SLBH$L_FLINK     EQ      0
                          031E          ASSUME  FWA$L_SLBH_FLINK+4 EQ FWA$L_SLBH_BLINK
                          031E          ASSUME  FWA$L_SLBH_FLINK-4 EQ FWA$L_SLB_PTR
                          031E          ASSUME  FWA$L_SLBH_FLINK-8 EQ FWA$L_SLBH_PTR
                          031E
                FFFFFFF8  031E  SLBH_PTR=-8                                     ; offset from R1 to slbh_ptr
                FFFFFFFC  031E  SLB_PTR=-4                                      ; offset from R1 to slb_ptr
                00000000  031E  SLBH_FLINK=0                                    ; offset from R1 to flink
                00000004  031E  SLBH_BLINK=4                                    ; offset from R1 to blink
                          031E
        00B0 CA     D0    031E          MOVL    FWA$L_SLBH_FLINK(R10),-
        00A8 CA           0322                  FWA$L_SLBH_PTR(R10)             ; init the 'current' SLBH
                          0325
                          0325  PRS_LOOP:
   51   00B0 CA     DE    0325          MOVAL   FWA$L_SLBH_FLINK(R10),R1        ; get que header addr
   50     F8 A1     D0    032A          MOVL    SLBH_PTR(R1),R0                 ; get next entry
     51      50     D1    032E          CMPL    R0,R1                           ; back to header?
             7D     13    0331          BEQL    CHKDIR                          ; done all list elements
   56     0B A0     9A    0333          MOVZBL  SLBH$B_STR_LEN(R0),R6           ; length of string
   57     14 A0     DE    0337          MOVAL   SLBH$T_STRING(R0),R7            ; addr of string
             19     8A    033B          BICB2   #FWA$C_ALLPASS,-
             6A           033D                  FWA$B_PASSFLGS(R10)             ; clear per pass flags
          0A A0     88    033E          BISB2   SLBH$B_PASSFLGS(R0),-
             6A           0341                  FWA$B_PASSFLGS(R10)             ; set current per pass flags
 FC A1    0C A0     D0    0342          MOVL    SLBH$L_SLB_QUE(R0),SLB_PTR(R1)  ; set 'current' SLB
                          0347
                          0347  ;
                          0347  ; Apply system defaults unless node name was specified.
                          0347  ; Check is based on the fact that the SET DEFAULT string is ALWAYS
                          0347  ; the last entry in the SLBH queue.
                          0347  ;
   04 A1     50     D1    0347          CMPL    R0,SLBH_BLINK(R1)               ; is this the set default SLBH?
             08     12    034B          BNEQ    5$                              ; nope, continue
   72 6A     19     E0    034D          BBS     #FWA$V_NODE,(R10),ENDPRS        ; all set if node present
   5B 6A     0F     E0    0351          BBS     #FWA$V_DEVICE,(R10),CHKDIR      ; branch if device present
                          0355
                          0355  ;
                          0355  ; If this is the related filename specification,
                          0355  ; first check to see if the parse of the related file could supply
                          0355  ; any of the missing filename elements, and if not, avoid this step.
                          0355  ;
                          0355
   42 6A     03     E1    0355  5$:     BBC     #FWA$V_RLF_PASS,(R10),30$       ; branch if not rlf pass
   13 68     3D     E1    0359          BBC     #FAB$V_OFP+FOP,(R8),10$         ; branch if input file parse
                          035D
                          035D          ASSUME  FWA$V_NAME       EQ      FWA$V_TYPE+1
                          035D
03   6A   02     0C     ED    035D          CMPZV   #FWA$V_TYPE,#2,(R10),#3         ; either name or type missing?
             17     12    0362          BNEQ    15$                             ; branch if yes (RLF can provide)
```

```
                     02 AA    38    93  0364  1022            BITB    #WCNTV_MASK,FWA$B_WILDFLGS(R10) ; wild name, type or ver?
                              11    12  0368  1023            BNEQ    15$                             ; branch if yes (RLF can provide)
                     0D 6A    1C    E0  036A  1024            BBS     #FWA$V_WILD_DIR,(R10),15$       ; any wild directories ?
                              36    11  036E  1025            BRB     40$                             ; don't both with RLF pass - no
                                        0370  1026                                                    ;  additional fields can be filled i
                                        0370  1027 ;
                                        0370  1028 ; This is an input file parse.
                                        0370  1029 ;
                                        0370  1030
                     07 6A    19    E1  0370  1031 10$:       BBC     #FWA$V_NODE,(R10),15$               ; branch if node name missing
                                        0374  1032
                                        0374  1033            ASSUME  FWA$V_NAME       EQ      FWA$V_TYPE+1
                                        0374  1034            ASSUME  FWA$V_DIR        EQ      FWA$V_NAME+1
                                        0374  1035            ASSUME  FWA$V_DEVICE     EQ      FWA$V_DIR+1
                                        0374  1036
          0F    6A    04    0C    ED  0374  1037            CMPZV   #FWA$V_TYPE,#4,(R10),#15            ; dev, dir, name or type missing?
                              2B    13  0379  1038            BEQL    40$                                 ; branch if not (RLF can't help)
                                        037B  1039
                                        037B  1040 ;
                                        037B  1041 ; At last we have a related file name string; parse it; but, before we
                                        037B  1042 ; do the related-file parse, make sure if the parse is of an output-file,
                                        037B  1043 ; that the filename and file type, if they are wild, are in the legal
                                        037B  1044 ; format (i.e. * and * only).
                                        037B  1045 ;
                                        037B  1046
                     1C 68    3D    E1  037B  1047 15$:       BBC     #FAB$V_OFP+FOP,(R8),30$             ; skip checks if input-file
                     0A 6A    15    E1  037F  1048            BBC     #FWA$V_WC_NAME,(R10),20$            ; skip check if not wild
                54     0170 8F    3C  0383  1049            MOVZWL  #FWA$Q_NAME,R4                      ; load offset to name descriptor
                            0BE7    30  0388  1050            BSBW    CHECK_WLD_FIELD                     ; if filename is of a legal wild
                              3A    12  038B  1051            BNEQ    WLDFNM                              ;  format, continue
                                        038D  1052
                     0A 6A    14    E1  038D  1053 20$:       BBC     #FWA$V_WC_TYPE,(R10),30$            ; skip check if type not wild
                54     0178 8F    3C  0391  1054            MOVZWL  #FWA$Q_TYPE,R4                      ; load offset to file type desc
                            0BD9    30  0396  1055            BSBW    CHECK_WLD_FIELD                     ; if file type is of a legal
                              32    12  0399  1056            BNEQ    WLDTYP                              ;  wild format continue
                                        039B  1057
    00B8 CA    18    00    6E    00    2C  039B  1058 30$:       MOVC5   #0,(SP),#0,#SLB$C_BLN,FWA$T_SLB(R10) ; clear fake SLB
                            002D    30  03A3  1059            BSBW    PARSE_STRING                        ; parse the string
                                        03A6  1060
                                        03A6  1061 ;
                                        03A6  1062 ; Since PARSE_STRING returned, the filename is not fully qualified.
                                        03A6  1063 ; Therefore, defaults must be applied.
                                        03A6  1064 ;
                                        03A6  1065
          00A8 CA    00A8 DA    D0  03A6  1066 40$:       MOVL    @FWA$L_SLBH_PTR(R10),FWA$L_SLBH_PTR(R10) ; advance to next
                            FF75    31  03AD  1067            BRW     PRS_LOOP                            ; and loop
```

```
                        03B0  1069 ;
                        03B0  1070 ; The filename string still is not fully qualified.
                        03B0  1071 ; Our last hope is to apply the default directory.
                        03B0  1072 ;
                        03B0  1073 ; Note: It is not necessary to upcase and compress the default directory string
                        03B0  1074 ;       because RMS$SETDD calls RM$XPFN to parse a new string prior to storing
                        03B0  1075 ;       it in PIO$GT_DDSTRING, thereby assuring that the default directory
                        03B0  1076 ;       string has been upcased and compressed.
                        03B0  1077 ;
                        03B0  1078
      0F 6A   0E    E0  03B0  1079 CHKDIR: BBS     #FWA$V_DIR,(R10),ENDPRS          ; branch if directory present
         6A   01    C8  03B4  1080         BISL2   #FWA$M_DUPOK,(R10)               ; allow duplicates
57  00000000'9F    9E  03B7  1081         MOVAB   @#PIO$GT_DDSTRING,R7             ; get address and size of default
         56   87    9A  03BE  1082         MOVZBL  (R7)+,R6                         ;  directory string
              10    10  03C1  1083         BSBB    PARSE_STRING                     ; parse default directory string
                        03C3  1084
                        03C3  1085 ;
                        03C3  1086 ; End of parse - not necessarily all elements present,
                        03C3  1087 ; but at least problem free
                        03C3  1088 ;
                        03C3  1089
                        03C3  1090 ENDPRS: RMSSUC                                   ; exit with success
                    05  03C6  1091         RSB
                        03C7  1092
                        03C7  1093 ;
                        03C7  1094 ; No partial wildcards allowed.
                        03C7  1095 ;
                        03C7  1096
                        03C7  1097 WLDFNM: RMSERR  FNM                              ; return the filename error
                    05  03CC  1098         RSB                                      ;
                        03CD  1099
                        03CD  1100 WLDTYP: RMSERR  TYP                              ; return the filetype error
                    05  03D2  1101         RSB
                        03D3  1102
```

```
03D3            .SBTTL  PARSE_STRING, Parse Individual Filename String
03D3
03D3    ;++
03D3    ;
03D3    ; PARSE_STRING - is the intermediate level filename string parser.
03D3    ;
03D3    ;       It accepts the particular string to be parsed as input from
03D3    ;       EXPAND_NAME and breaks it up into its constituent filename
03D3    ;       elements and either saves these (by moving them to the
03D3    ;       appropriate element buffer) or discards them if they are
03D3    ;       permissable duplicates.
03D3    ;
03D3    ;       If a node spec is seen, an immediate attempt is made to translate the
03D3    ;       nodename portion of the string. If this succeeds and the equivalence
03D3    ;       string consists of one or more concatenated node specs, then the
03D3    ;       left-most node spec is isolated and the process is repeated until
03D3    ;       logical node name translation fails. Then, the primary node spec is
03D3    ;       moved to the node name element buffer and all secondary node specs are
03D3    ;       are appended to it. Also, there are rules for defaulting access control
03D3    ;       information with respect to the recursive logical node name translation
03D3    ;       procedure.
03D3    ;
03D3    ;       If a logical/device name is seen, it is handled as follows:
03D3    ;       Initially it is not moved; instead a descriptor is set-up to point
03D3    ;       to it. After parsing all remaining elements of the string, an attempt
03D3    ;       is made to translate the logical/device name as a logical name. If
03D3    ;       this succeeds, the equivalence string is then in turn parsed and its
03D3    ;       elements merged in or discarded as for the original string. If the
03D3    ;       translation fails, the equivalence string (which is identical to the
03D3    ;       logical/device name string minus a leading underscore, if any) is taken
03D3    ;       as a physical device name and processed accordingly.
03D3    ;
03D3    ;       A logical/filename is treated similarly, with the following differences:
03D3    ;       It is the only string element hence there is no need to continue
03D3    ;       parsing the remainder of the input string. Moreover, if the translation
03D3    ;       does not succeed the equivalence string is treated as a filename.
03D3    ;
03D3    ;       The general flow of control of this routine is as follows:
03D3    ;
03D3    ;
03D3    ;                 XPFN
03D3    ;                   |       ^           ^
03D3    ;                   |       |error      |
03D3    ;                   |       |success    |
03D3    ;               EXPAND_NAME--           |
03D3    ;                   |       ^           |
03D3    ;                   |       |           |error
03D3    ;                   |       |success    |parse done
03D3    ;               PARSE_STRING-------------
03D3    ;                   |       ^           ^          NOTE: Some work routines called
03D3    ;                   |       |error      |                by parse_string pop the
03D3    ;                   |       |success    |                stack so they can operate
03D3    ;               work_routine-           |                at the level of parse_string
03D3    ;                   |       ^           |
03D3    ;                   |       |           |
03D3    ;                   |       |success    |error
03D3    ;               CHECK_FIELD--------------
```

```
                        03D3  1161 ;
                        03D3  1162 ; Calling Sequence:
                        03D3  1163 ;
                        03D3  1164 ;       BSBW    PARSE_STRING
                        03D3  1165 ;
                        03D3  1166 ; Input Parameters:
                        03D3  1167 ;
                        03D3  1168 ;       R6      Filename string length
                        03D3  1169 ;       R7      Filename string address
                        03D3  1170 ;       R10     FWA address
                        03D3  1171 ;       R11     FSCB address (there are two)
                        03D3  1172 ;
                        03D3  1173 ; Implicit Inputs:
                        03D3  1174 ;
                        03D3  1175 ;       The current contents of the FWA.
                        03D3  1176 ;       The contents of the filename string input.
                        03D3  1177 ;       The contents of the logical name tables
                        03D3  1178 ;
                        03D3  1179 ; Outputs:
                        03D3  1180 ;
                        03D3  1181 ;       R0-R7   Destroyed
                        03D3  1182 ;       AP      Destroyed
                        03D3  1183 ;
                        03D3  1184 ; Implicit Outputs:
                        03D3  1185 ;
                        03D3  1186 ;       The FWA is updated as per the filename element seen
                        03D3  1187 ;
                        03D3  1188 ; Completion Codes:
                        03D3  1189 ;
                        03D3  1190 ;       PARSE_STRING returns to the caller only if there were no errors
                        03D3  1191 ;       encountered and there remain one or more missing filename elements
                        03D3  1192 ;       (node excluded). If either or these occur the return PC is popped
                        03D3  1193 ;       from the stack and, if an error, an RSB is performed with the status
                        03D3  1194 ;       code in R0 (standard RMS), else a branch to ENDPRS is taken.
                        03D3  1195 ;
                        03D3  1196 ; Side Effects:
                        03D3  1197 ;
                        03D3  1198 ;       None
                        03D3  1199 ;
                        03D3  1200 ;--
                        03D3  1201
                        03D3  1202 PARSE_STRING:
            56     DD   03D3  1203         PUSHL   R6                              ; save lehgth
         FC28'     30   03D5  1204         BSBW    RM$SCAN_STRING                  ; scan string for elements
            56   8ED0   03D8  1205         POPL    R6                              ; restore length
  04 AB     56     B1   03DB  1206         CMPW    R6,FSCB$Q_FILESPEC(R11)         ; the lengths must match
            08     13   03DF  1207         BEQL    30$                             ; they do
                        03E1  1208         RMSERR  SYN                             ; syntax error
          0154     31   03E6  1209 20$:    BRW     PRS_ERROR                       ; exit with error
                        03E9  1210
                        03E9  1211         ASSUME  FSCB$B_FLDFLAGS EQ      0
                        03E9  1212
     6B     10     91   03E9  1213 30$:    CMPB    #FSCB$M_NAME,(R11)              ; do we have only a name
            07     12   03EC  1214         BNEQ    40$
  56     2C AB     7D   03EE  1215         MOVQ    FSCB$Q_NAME(R11),R6             ; get the descriptor
          014C     31   03F2  1216         BRW     FILE_OR_LOGNAME                 ; could be name or logname
                        03F5  1217
```

```
11 6B    00   E1  03F5   1218 40$:      BBC     #FSCB$V_NODE,(R11),50$          ; is there a node spec
56    44 AB   7D  03F9   1219           MOVQ    FSCB$Q_NODE1(R11),R6
       0180   30  03FD   1220           BSBW    CHECK_NODE
      07 50   E8  0400   1221           BLBS    R0,50$
                  0403   1222           RMSERR  NOD
         DC   11  0408   1223           BRB     20$
                  040A   1224
13 6B    01   E1  040A   1225 50$:      BBC     #FSCB$V_DEVICE,(R11),60$        ; is there a device name
56    14 AB   7D  040E   1226           MOVQ    FSCB$Q_DEVICE(R11),R6
         56   B7  0412   1227           DECW    R6                              ; remove trailing ':'
       0156   30  0414   1228           BSBW    CHECK_DEVICE
      07 50   E8  0417   1229           BLBS    R0,60$
                  041A   1230           RMSERR  DEV
         C5   11  041F   1231           BRB     20$
                  0421   1232
11 6B    03   E1  0421   1233 60$:      BBC     #FSCB$V_DIRECTORY,(R11),65$     ; is there a directory spec
56    24 AB   7D  0425   1234           MOVQ    FSCB$Q_DIRECTORY(R11),R6
       047B   30  0429   1235           BSBW    PARSE_DIR                       ; go parse directory spec
      07 50   E8  042C   1236           BLBS    R0,65$
                  042F   1237           RMSERR  DIR
         B0   11  0434   1238           BRB     20$
                  0436   1239
11 6B    02   E1  0436   1240 65$:      BBC     #FSCB$V_ROOT,(R11),70$          ; is there a root directory spec
                  043A   1241           SSB     #FWA$V_EXP_ROOT,(R10)           ; flag that it was explicit
       0809   30  043E   1242           BSBW    CHECK_ROOT                      ; go parse root directory spec
      07 50   E8  0441   1243           BLBS    R0,70$
                  0444   1244           RMSERR  DIR
         3B   11  0449   1245           BRB     99$
                  044B   1246
11 6B    04   E1  044B   1247 70$:      BBC     #FSCB$V_NAME,(R11),80$          ; is there a file name
56    2C AB   7D  044F   1248           MOVQ    FSCB$Q_NAME(R11),R6
       0411   30  0453   1249           BSBW    CHECK_NAME
      07 50   E8  0456   1250           BLBS    R0,80$
                  0459   1251           RMSERR  FNM
         26   11  045E   1252           BRB     99$
                  0460   1253
10 6B    05   E1  0460   1254 80$:      BBC     #FSCB$V_TYPE,(R11),90$          ; is there a file type
56    34 AB   7D  0464   1255           MOVQ    FSCB$Q_TYPE(R11),R6
         22   10  0468   1256           BSBB    CHECK_TYPE
      07 50   E8  046A   1257           BLBS    R0,90$
                  046D   1258           RMSERR  TYP
         12   11  0472   1259           BRB     99$
                  0474   1260
11 6B    06   E1  0474   1261 90$:      BBC     #FSCB$V_VERSION,(R11),100$      ; is there a version number
56    3C AB   7D  0478   1262           MOVQ    FSCB$Q_VERSION(R11),R6
         2C   10  047C   1263           BSBB    CHECK_VERSION
      08 50   E8  047E   1264           BLBS    R0,100$
                  0481   1265           RMSERR  VER
       00B4   31  0486   1266 99$:      BRW     PRS_ERROR                       ; an error
                  0489   1267
       003C   31  0489   1268 100$:     BRW     FINISH_PARSE                    ; go finish up
```

```
                        048C  1270 ;
                        048C  1271 ; File type
                        048C  1272 ;
                        048C  1273
                        048C  1274 CHECK_TYPE:
   04 6A     04    E1   048C  1275          BBC     #FWA$V_FNA_PASS,(R10),10$; branch if not FNA pass
                        0490  1276          SSB     #FWA$V_EXP_TYPE,(R10)     ; set explicit version flag
             56    B7   0494  1277 10$:     DECW    R6                        ; remove the leading terminator
             57    D6   0496  1278          INCL    R7                        ;
        50   0C    9A   0498  1279          MOVZBL  #FWA$V_TYPE,R0            ; flag field
        51   28    9A   049B  1280          MOVZBL  #FWA$C_MAXTYPE+1,R1       ; check the length
 55   0178 CA      7E   049E  1281          MOVAQ   FWA$Q_TYPE(R10),R5        ; get output descriptor
           08DF    30   04A3  1282          BSBW    CHECK_FIELD               ; process field
                        04A6  1283          RMSSUC
                   05   04A9  1284          RSB
```

```
                         04AA  1286 ;
                         04AA  1287 ; File version
                         04AA  1288 ;
                         04AA  1289
                         04AA  1290 CHECK_VERSION:
    04 6A    04    E1    04AA  1291            BBC     #FWA$V_FNA_PASS,(R10),10$; branch if not FNA pass
                         04AE  1292            SSB     #FWA$V_EXP_VER,(R10)     ; set explicit version flag
             56    B7    04B2  1293 10$:       DECW    R6                       ; remove the leading terminator
             57    D6    04B4  1294            INCL    R7                       ;
       50    0B    9A    04B6  1295            MOVZBL  #FWA$V_VERSION,R0        ; flag field
       51    07    9A    04B9  1296            MOVZBL  #FWA$C_MAXVER+1,R1       ; length
 55    0180  CA    7E    04BC  1297            MOVAQ   FWA$Q_VERSION(R10),R5    ; get output descriptor
           08C1    30    04C1  1298            BSBW    CHECK_FIELD              ; process field
                         04C4  1299            RMSSUC
             05          04C7  1300            RSB
```

```
                         04C8    :::
                         04C8    ::: We have both a logical name and a device;
                         04C8    ::: ignore the logical name unless no duplicates allowed.
                         04C8    :::
                         04C8    :::
                         04C8    :::devdup:      bbs     #fwa$v_dupok,(r10),prs_exit
                         04C8    :::       rmserr  dev
                         04C8
                         04C8    ;
                         04C8    ; The filename string has been parsed into its basic parts.
                         04C8    ; Check to see if we got a logical name, and if so go attempt
                         04C8    ; to translate it.
                         04C8    ;
                         04C8
                         04C8    FINISH_PARSE:
      28 6A   30   E5    04C8            BBCC    #FWA$V_LOGNAME,(R10),PRS_EXIT   ; Branch if no logical name
   56   00D0 CA   7D     04CC            MOVQ    FWA$Q_LOGNAM(R10),R6            ; Get descriptor of name
      03 6A   19   E0    04D1            BBS     #FWA$V_NODE,(R10),15$           ; Branch if node spec -
                         04D5                                                    ; don't translate device names
                         04D5    ;
                         04D5    ; In order for searchlists to be completely evaluated, it is necessary
                         04D5    ; that they be translated, even if a device-name is already present.
                         04D5    ; Only if they don't translate and try to actually get used as a device
                         04D5    ; name are they actually not-used.   The CHECK_FIELD subroutine
                         04D5    ; properly knows how to handle this, in particular, allowing the translation
                         04D5    ; of the FNA string to yield a duplicate device if the NAM_DVI device
                         04D5    ; was used.
                         04D5    ;
                         04D5    ::: This is a change from pre-V4 where logical devices names were ignored
                         04D5    ::: if a device name already existed.  This is viewed as a ''bug'' in the
                         04D5    ::: original implementation but it's consequences aren't yet fully understood.
                         04D5    ::: If all goes well with making this change, then these 2 lines may be deleted.
                         04D5    ::: Other wise they must be restored and the 2nd ''10$'' replaced by 'DEVDUP''.
                         04D5    ::: See also the code at DEVDUP and FILE_OR_LOGNAME.
                         04D5    :::
                         04D5    :::
                         04D5    :::     bbs     #fwa$v_nam_dvi,(r10),10$        ; Branch if device came from NAM blo
                         04D5    :::                                             ;  (i.e., do the translation)
                         04D5    :::     bbs     #fwa$v_device,(r10),10$         ; Branch if device already seen
                         04D5
                         04D5    ;
                         04D5    ; Go attempt to translate the name.
                         04D5    ; If no translation return here and treat as device name.
                         04D5    ;
                         04D5
            05CB   30    04D5    10$:    BSBW    TRANSLATE
                         04D8
                         04D8    ;
                         04D8    ; It appears as if it was a device name after all.  Allow the name even if
                         04D8    ; not in standard device name format - if truly invalid it will be caught when
                         04D8    ; attempting to assign the channel.  This allows flexibility for different
                         04D8    ; device/controller/unit naming schemes as well as remote node logical names.
                         04D8    ;
                         04D8
                         04D8    ;
                         04D8    ; In order to call check_field (which would return one level up to EXPAND_NAME
                         04D8    ; on an error) we must make this code run at work routine level.  This can only
```

```
                     04D8   1359 ; return on and error from check_field
                     04D8   1360 ;
                     04D8   1361
        08     10    04D8   1362 15$:     BSBB    20$
                     04DA   1363          RMSERR  DEV
      005B     31    04DF   1364          BRW     PRS_ERROR
                     04E2   1365
    50    0F   9A    04E2   1366 20$:     MOVZBL  #FWA$V_DEVICE,R0                    ; flag which field
  51   FF 8F   9A    04E5   1367          MOVZBL  #FWA$S_DEVICEBUF,R1                 ; length
55   00E0 CA   7E    04E9   1368          MOVAQ   FWA$Q_DEVICE(R10),R5                ; output descriptor
      0894     30    04EE   1369          BSBW    CHECK_FIELD
                     04F1   1370 ;;;      blbc    r0,30$                              ; only if we used this spec.
                     04F1   1371 ;;;      cmpb    @fwa$q_lognam+4(r10),#^a/_/         ; is 1st char an underscore?
                     04F1   1372 ;;;      bneq    30$                                 ; branch if not
                     04F1   1373 ;;;      ssb     #fwa$v_dev_under,(r10)              ; flag leading underscore
        50 8ED0      04F1   1374 30$:     POPL    R0                                  ; pop return pc and exit
```

```
                        04F4  1376 ;
                        04F4  1377 ; PRS_EXIT ...
                        04F4  1378 ;
                        04F4  1379 ; The parse of this string is complete.
                        04F4  1380 ;
                        04F4  1381 ; If this was the parse of the primary name string (FNA), handle explicit
                        04F4  1382 ; flags. In particular, use the explicit type and version flags to
                        04F4  1383 ; set the type and version seen flags (i.e., explicit null type and
                        04F4  1384 ; version cause no defaults to be taken for these fields).  Also,
                        04F4  1385 ; copy the current device and directory seen flags to the explicit
                        04F4  1386 ; device and directory.
                        04F4  1387 ;
                        04F4  1388
                        04F4  1389 PRS_EXIT:
       22 6A    04    E1 04F4  1390         BBC     #FWA$V_FNA_PASS,(R10),20$       ; branch if not FNA pass
                        04F8  1391
                        04F8  1392         ASSUME  FWA$V_EXP_VER    EQ      <FWA$B_WILDFLGS*8>
                        04F8  1393         ASSUME  FWA$V_EXP_TYPE   EQ      FWA$V_EXP_VER+1
                        04F8  1394         ASSUME  FWA$V_TYPE       EQ      FWA$V_VERSION+1
                        04F8  1395
                        04F8  1396 ;
                        04F8  1397 ; If EXP_TYPE is not set but TYPE is set, then its ok, don't unconditionally
                        04F8  1398 ; overwrite TYPE.
                        04F8  1399 ;
                        04F8  1400
          02 AA       F0 04F8  1401         INSV    FWA$B_WILDFLGS(R10),-           ; set version from
    6A    01  0B         04FB  1402                 #FWA$V_VERSION,#1,(R10)         ;  'explicit' flag
       06 6A    11    E1 04FE  1403         BBC     #FWA$V_EXP_TYPE,(R10),10$       ; if EXP_TYPE not set,
                        0502  1404                                                 ;  then ignore TYPE
          02 AA       F0 0502  1405         INSV    FWA$B_WILDFLGS(R10),-           ; set version and type from
    6A    02  0B         0505  1406                 #FWA$V_VERSION,#2,(R10)         ;  'explicit' flags
                        0508  1407
                        0508  1408         ASSUME  FWA$V_EXP_DIR    EQ      FWA$V_EXP_DEV-1
                        0508  1409         ASSUME  FWA$V_DIR        EQ      FWA$V_DEVICE-1
                        0508  1410
50    6A    02    0E  EF 0508  1411 10$:    EXTZV   #FWA$V_DIR,#2,(R10),R0          ; get dev and dir
6A    02    16    50  F0 050D  1412         INSV    R0,#FWA$V_EXP_DIR,#2,(R10)      ; and set explicit dev & dir
       0C 6A    19    E1 0512  1413         BBC     #FWA$V_NODE,(R10),30$           ; if not network parse,
                        0516  1414         SSB     #FWA$V_EXP_NODE,(R10)           ; mark explicit node
                        051A  1415
                        051A  1416 ;
                        051A  1417 ; If we have a node name and quoted string, then stop the parse in its tracks
                        051A  1418 ; and don't go looking for trouble
                        051A  1419 ;
                        051A  1420
       04 6A    19    E1 051A  1421 20$:    BBC     #FWA$V_NODE,(R10),30$           ; if not network parse,
                        051E  1422                                                 ;  then skip it
       15 6A    1A    E0 051E  1423         BBS     #FWA$V_QUOTED,(R10),PRS_EXIT1   ; if quoted then stop parsing
                        0522  1424
                        0522  1425 ;
                        0522  1426 ; Check for a fully qualified file name string.
                        0522  1427 ; If any fields are missing, return to caller to apply defaults.
                        0522  1428 ;
                        0522  1429
    F8 8F    01 AA    91 0522  1430 30$:    CMPB    FWA$B_FLDFLGS(R10),#FWA$C_ALL   ; all of dev,dir,nam,typ,ver
                10    12 0527  1431         BNEQ    PRS_EXIT2                       ; branch if not
       0A 68    3D    E1 0529  1432         BBC     #FAB$V_OFP+FOP,(R8),PRS_EXIT1   ; branch if input file parse
```

```
  02 AA    38    93  052D   1433            BITB    #WCNTV_MASK,FWA$B_WILDFLGS(R10) ; wild name, type, or ver?
           06    12  0531   1434            BNEQ    PRS_EXIT2                       ; if yes, go apply defaults
     6A    1C    E0  0533   1435            BBS     #FWA$V_WILD_DIR,(R10),-         ; if any wild directories
           02        0536   1436                    PRS_EXIT2                       ;  go apply defaults
                     0537   1437
                     0537   1438 ;
                     0537   1439 ; return to stop parsing
                     0537   1440 ;
                     0537   1441
                     0537   1442 PRS_EXIT1:
           01    BA  0537   1443            POPR    #^M<R0>                         ; pop return PC
                     0539   1444
                     0539   1445 ;
                     0539   1446 ; return to apply defaults
                     0539   1447 ;
                     0539   1448
                     0539   1449 PRS_EXIT2:
                     0539   1450            RMSSUC                                  ; exit from parse successfully
                 05  053C   1451            RSB
                     053D   1452
                     053D   1453 ;
                     053D   1454 ; An error condition has been encountered.
                     053D   1455 ; Error code is in R0 - pop the extra PC from the stack and return
                     053D   1456 ;
                     053D   1457
                     053D   1458 PRS_ERROR:
           51  8ED0  053D   1459            POPL    R1
                 05  0540   1460            RSB
                     0541   1461
                 0541      ;
                 0541      ; We have a single field with no punctuation; it is either a
                 0541      ; logical name or a file name. In order for it to be a logical name
                 0541      ; the following must be true:
                 0541      ;
                 0541      ;        1.  there must be no other filename elements.
                 0541      ;        2.  it must translate.
                 0541      ;
                 0541      ; Previous to V4, as described at FINISH_PARSE, once a device name was
                 0541      ; seen then all device/logical names were ignored and a file/logical name
                 0541      ; was forced to be treated as a filename.   With the advent of search lists,
                 0541      ; this is no longer true, and a file/logical name is ALWAYS attempted to
                 0541      ; be translated.  The commented-out code may be removed at the same time
                 0541      ; as the FINISH_PARSE code.
                 0541      ;
                 0541
                 0541      FILE_OR_LOGNAME:
                 0541      ;;;       tstb     fwa$b_fldflgs(r10)       ; any flags set?
                 0541      ;;;       beql     10$
                 0541      ;;;
                 0541      ;;;       ASSUME   FWA$V_DEVICE     GE       8
                 0541      ;;;
                 0541      ;;;       cmpb     fwa$b_fldflgs(r10),-     ; only device seen
                 0541      ;;;                #<fwa$m_device@-8>
                 0541      ;;;       bneq     30$
                 0541      ;;;       bbc      #fwa$v_nam_dvi,(r10),30$; branch, testing this as a filename
                 0541      ;;;                                         ;  unless device came from NAM block
18 6A   19   E0  0541      10$:      BBS      #FWA$V_NODE,(R10),30$    ; branch if node seen
   2C AB     7D  0545                MOVQ     FSCB$Q_NAME(R11),-       ; store the name to translate
 00D0 CA         0548                         FWA$Q_LOGNAM(R10)
                 054B
                 054B      ;
                 054B      ; Try to translate the name, if it is a logical name then translate does not
                 054B      ; return but continues parsing the new logical name.  If it is NOT a logical
                 054B      ; name -or- the translation is not to be used then it exits with R0
                 054B      ; indicating which it was.
                 054B      ;
                 054B
     0555    30  054B                BSBW     TRANSLATE                        ; attempt translation, if translate
                 054E                                                          ;  returns failure, its a filename
   0C 50     E9  054E                BLBC     R0,30$                           ; was this a hidden device
      1A     10  0551                BSBB     CHECK_DEVICE                     ; yes, treat as a device name
   14 50     E8  0553                BLBS     R0,40$                           ; was there an error
                 0556                RMSERR   DEV                              ; call it a device error
      E0     11  055B                BRB      PRS_ERROR                        ; exit
                 055D
                 055D      ;
                 055D      ; Say result name is really the filename
                 055D      ;
                 055D
     0307    30  055D      30$:      BSBW     CHECK_NAME                       ; process it that way
   07 50     E8  0560                BLBS     R0,40$                           ; was there an error
                 0563                RMSERR   FNM                              ; call it a file name error
      D3     11  0568                BRB      PRS_ERROR                        ; exit
                 056A
     FF5B    31  056A      40$:      BRW      FINISH_PARSE                     ; finish up parsing
```

```
                     056D  1520 ;
                     056D  1521 ; Process logical/device name
                     056D  1522 ;
                     056D  1523
                     056D  1524 CHECK_DEVICE:
      50    30    9A 056D  1525         MOVZBL  #FWA$V_LOGNAME,R0          ; flag field
   51    FF 8F    9A 0570  1526         MOVZBL  #LNM$C_NAMLENGTH,R1        ; length
55    00D0 CA    7E 0574  1527         MOVAQ   FWA$Q_LOGNAM(R10),R5       ; get output descriptor
         0809    30 0579  1528         BSBW    CHECK_FIELD                ; process field
                     057C  1529         RMSSUC
                  05 057F  1530 20$:    RSB
                     0580  1531
```

```
                       0580  1533          .SBTTL  CHECK_NODE, Check Node Specification
                       0580  1534  ;
                       0580  1535  ; We have a primary node spec...
                       0580  1536  ;
                       0580  1537  ; A file specification string may begin with a list of concatenated
                       0580  1538  ; node spec strings (i.e., node1::node2::node3::dev:[dir]file.typ;ver).
                       0580  1539  ; Further, the first non-null node spec is a candidate for iterative
                       0580  1540  ; logical node name translation, where each equivalence string may yield one
                       0580  1541  ; or more concatenated node spec strings!
                       0580  1542  ;
                       0580  1543
                       0580  1544  CHECK_NODE:
                       0580  1545
                       0580  1546  ;
                       0580  1547  ; Since the first node spec can only be null we can check here
                       0580  1548  ;
                       0580  1549
    08 56    14    E1  0580  1550          BBC     #FSCB$V_NULL,R6,10$         ; branch if not null
                       0584  1551          SSB     #FWA$V_EXP_NODE,(R10)       ; set flag to disable stickness
                       0588  1552          RMSSUC
                   05  058B  1553  5$:     RSB                                 ; continue with next field
                       058C  1554
                       058C  1555  ;
                       058C  1556  ; Allocate space for the Network Work Area (NWA) unless it has already
                       058C  1557  ; been allocated via previous pass thru this code.
                       058C  1558  ;
                       058C  1559
       3C A9       D5  058C  1560  10$:    TSTL    IFB$L_NWA_PTR(R9)           ; if nwa there?
          12       12  058F  1561          BNEQ    20$                         ; branch if yes
          5B       DD  0591  1562          PUSHL   R11                         ; save FSCB address
       54 AA       D0  0593  1563          MOVL    FWA$L_IMPURE_AREA(R10),-; restore impure area pointer
          5B           0596  1564                  R11
   00000000'EF     16  0597  1565          JSB     NT$NWA_INIT                 ; allocate and initialize NWA
          5B     8ED0  059D  1566          POPL    R11                         ; restore FSCB
       E8 50       E9  05A0  1567          BLBC    R0,5$                       ; branch for error exit
                       05A3  1568
```

```
                        05A3  ;
                        05A3  ; Logical node name translation...
                        05A3  ;
                        05A3  ; Attempt to translate the node name string of the (left-most) node spec.
                        05A3  ; For the translation to succeed, the equivalence string must represent
                        05A3  ; one or more concatenated node specs, i.e., each field must be delimited
                        05A3  ; by a double colon.
                        05A3  ;
                        05A3  ; Note: If node name is prefixed by underscore, no translation occurs
                        05A3  ;       but the underscore is removed.
                        05A3  ;
                        05A3  ; Note: If node name contains an access control string with the password masked
                        05A3  ;       out, then an attempt is made to translate the string consisting of
                        05A3  ;       <node name + access control string> as a special logical node name.
                        05A3  ;
                        05A3
                        05A3          ASSUME  FWA$T_NODEBUF   EQ      <FWA$B_UNDER_NOD+1>
                        05A3
                59  DD  05A3  20$:    PUSHL   R9                      ; save ifb addr
      59    3C  A9  D0  05A5          MOVL    IFB$L_NWA_PTR(R9),R9    ; get address of NWA
          0244  C9  D4  05A9          CLRL    NWA$Q_ACS(R9)           ; zero ACS size/flag
          025C  C9  D4  05AD          CLRL    NWA$Q_INODE(R9)         ; initialize intermediate node spec desc
          052C  C9  9E  05B1          MOVAB   NWA$T_INODEBUF+NWA$C_INODESIZ(R9),-
          0260  C9      05B5                  NWA$Q_INODE+4(R9)       ; strings will be added
                        05B8                                          ;  from right to left
                        05B8  TRANSLATE_LOOP:
     0F 56      15  E1  05B8          BBC     #FSCB$V_PWD,R6,10$      ; branch if password is not masked out
        56      02  A2  05BC          SUBW2   #2,R6                   ;  of access control string of node spec
     67     5F  8F  91  05BF          CMPB    #^A/_/,(R7)             ; remove leading underscore character
                28  12  05C3          BNEQ    20$                     ;  (if present) before attempting
                56  D7  05C5          DECL    R6                      ;  translation of special logical node
                57  D6  05C7          INCL    R7                      ;  name string with password masked out
                22  11  05C9          BRB     20$                     ; translate
                        05CB
              0236  30  05CB  10$:    BSBW    PSEUDO_CHKFLD           ; validate node spec syntax
        56      02  A2  05CE          SUBW2   #2,R6                   ; remove trailing '::'
     18 56      12  E1  05D1          BBC     #FSCB$V_ACS,R6,20$      ; branch if there is no embedded access
                        05D5                                          ;  control string in node spec string
    67  56      22  3A  05D5          LOCC    #^A/"/,R6,(R7)          ; find start of access control string
                        05D9                                          ;  on return <R0,R1> => ACS
        56      50  A2  05D9          SUBW2   R0,R6                   ; compute size of node name string
          0244  C9  D5  05DC          TSTL    NWA$Q_ACS(R9)           ; ignore this access control string if
                0B  12  05E0          BNEQ    20$                     ;  there is a default one in ACS buffer
     0244 C9    50  D0  05E2          MOVL    R0,NWA$Q_ACS(R9)        ; store size of ACS
0248 D9  61     50  28  05E7          MOVC3   R0,(R1),@NWA$Q_ACS+4(R9); make this current default ACS string
     023C C9    56  7D  05ED  20$:    MOVQ    R6,NWA$Q_LOGNAME(R9)    ; store descriptor of node name string
                        05F2                                          ;  to translate
                        05F2
                09  E0  05F2          BBS     #LNM$V_TERMINAL,-       ; Marked terminal?
        5B 0238 C9      05F4                  NWA$L_XLTATTR(R9),47$   ; Branch if so
                        05F8
                        05F8          ASSUME  NWA$L_XLTBUFFLG EQ      NWA$L_XLTCNT+4
                        05F8
     51   0228  C9  DE  05F8          MOVAL   NWA$L_XLTCNT(R9),R1     ; Get ptr to xlt count
                0A  F3  05FD          AOBLEQ  #LNM$C_MAXDEPTH,-
             09 81      05FF                  (R1)+,40$               ; Branch if < 10 translations done
             59 8ED0    0601  30$:    POPL    R9
             51 8ED0  0604          POPL    R1                        ; Discard CHECK_NODE return pc
        052F     31   0607          BRW     ERRLNE                    ; Declare error in logical name
                      060A
                      060A   ;
                      060A   ; Attempt logical node name translation.
                      060A   ;
                      060A
55   02AC C9     DE   060A   40$:   MOVAL   NWA$T_XLTBUF1(R9),R5      ; addr of buffer for translation
     81   61     D2   060F          MCOML   (R1),(R1)+                ; check if this buffer is free
          05     12   0612          BNEQ    45$                       ; branch if ok
55   03AC C9     DE   0614          MOVAL   NWA$T_XLTBUF2(R9),R5      ; swap buffers
0210 C9   55     D0   0619   45$:   MOVL    R5,NWA$T_ITM_STRING+4(R9); set address of buffer selected
     0A97'CF     9F   061E          PUSHAB  W^TABNAM                  ; make descriptor of table name
          0C'    DD   0622          PUSHL   S^#TABNAM_SIZE
     51   5E     D0   0624          MOVL    SP,R1                     ; remember where it is
                      0627          $TSTPT  NTXLATLOG                 ;
                      062D          $TRNLNM_S       -                 ; Translate the name
                      062D                  LOGNAM=NWA$Q_LOGNAME(R9),-
                      062D                  TABNAM=(R1),-
                      062D                  ATTR=W^LNM_ATTR,-
                      062D                  ITMLST=NWA$T_ITM_LST(R9),-
                      062D                  ACMODE=FWA$B_XLTMODE(R10)
     5E   08     C0   0646          ADDL2   #8,SP                     ; discard table name descriptor
        0F 50    E8   0649          BLBS    R0,50$                    ; Was translation successful?
01BC 8F   50     B1   064C          CMPW    R0,#SS$_NOLOGNAM          ; no such translation?
          AE     12   0651          BNEQ    30$                       ; quit on odd error
54   023C C9     7D   0653   47$:   MOVQ    NWA$Q_LOGNAME(R9),R4      ; get back source string descriptor
        00AF     31   0658          BRW     TRANSLATE_FAIL            ; proceed with source
                      065B
     0234 C9     D5   065B   50$:   TSTL    NWA$L_XLTMAXINDX(R9)      ; check for searchlists
          F2     19   065F          BLSS    47$                       ; no good ones, assume no tran
          42     14   0661          BGTR    ERRNOD                    ; no searchlists allowed
          0A     E1   0663          BBC     #LNM$V_EXISTS,-           ; must be a translation 0
  3C 0238 C9          0665                  NWA$L_XLTATTR(R9),ERRNOD
54   0230 C9     3C   0669          MOVZWL  NWA$L_XLTSIZ(R9),R4       ; Store equivalence string size
                      066E                                            ;  NOTE: <R4,R5> => equivalence string
                      066E
                      066E   ;
                      066E   ; Process the equivalence string.
                      066E   ;
                      066E
     56   54     7D   066E          MOVQ    R4,R6                     ; Copy descriptor to required registers
        06A1     30   0671          BSBW    UPCASE_COMPRESS           ; Perform upcasing and string
                      0674                                            ;  compression in place
5B   0104 CB     9E   0674          MOVAB   FSCB$K_BLN(R11),R11       ; use secondary FSCB
          56     DD   0679          PUSHL   R6                        ; Save length of input
        F982'    30   067B          BSBW    RM$SCAN_STRING
          56   8ED0   067E          POPL    R6                        ; Retrieve length of input
  04 AB   56     B1   0681          CMPW    R6,FSCB$Q_FILESPEC(R11)   ; Did we use all of it?
          1E     12   0685          BNEQ    ERRNOD                    ; nope, then return error
                      0687
                      0687          ASSUME  FSCB$B_FLDFLAGS EQ      0
                      0687
     6B   01     91   0687          CMPB    #FSCB$M_NODE,(R11)        ; there better be just nodes
          19     12   068A          BNEQ    ERRNOD
                      068C
                      068C   ;
                      068C  1684 ; Are thier intermediate node names?
                      068C  1685 ;
     50    01 AB   9A 068C  1686         MOVZBL  FSCB$B_NODES(R11),R0     ; get number of nodes
              50   D7 0690  1687         DECL    R0                       ; remove the first and if not
              02   13 0692  1688         BEQL    60$                      ;  more than one node spec string skip
              18   10 0694  1689         BSBB    INTERMED_NODE            ; process extra node names
     56    44 AB   7D 0696  1690 60$:    MOVQ    FSCB$Q_NODE1(R11),R6     ; get next node for translation
   5B    FEFC CB   9E 069A  1691         MOVAB   -FSCB$K_BLN(R11),R11     ; restore primary FSCB
            FF16   31 069F  1692         BRW     TRANSLATE_LOOP           ; try another translation
                      06A2  1693
                      06A2  1694 ;
                      06A2  1695 ; Handle node name errors.
                      06A2  1696 ;
                      06A2  1697
            50 8ED0   06A2  1698 ERRNOD1:POPL    R0                       ; discard return pc
                      06A5  1699 ERRNOD: RMSERR  NOD                      ; declare error in node name
            59 8ED0   06AA  1700         POPL    R9                       ; restore ifb
                 05   06AD  1701         RSB                              ; exit with completion code in R0
```

```
                                06AE   1703 ;
                                06AE   1704 ; Process 2nd to nth node specs found in the equivalence string.
                                06AE   1705 ; Node names in these node specs are not candidates for logical name
                                06AE   1706 ; translation. Also, any access control string supplied with the above
                                06AE   1707 ; logical name is the default (and overrides any) access control string
                                06AE   1708 ; for the last node spec of the equivalence string.
                                06AE   1709 ;
                                06AE   1710 ; Input:
                                06AE   1711 ;
                                06AE   1712 ;        R0       - Index of last node spec
                                06AE   1713 ;
                                06AE   1714
                                06AE   1715 INTERMED_NODE:
                                06AE   1716
                                06AE   1717 ;
                                06AE   1718 ; Handle defaulting of access control string and append 2nd to nth node specs
                                06AE   1719 ; to an intermediate node spec buffer.
                                06AE   1720 ;
                                06AE   1721
          57    50 AB     D0    06AE   1722          MOVL    FSCB$Q_NODE2+4(R11),R7  ; get address of rest of string
                44 AB     A3    06B2   1723          SUBW3   FSCB$Q_NODE1(R11),-     ; get size of string minus the
          56    0C AB           06B5   1724                  FSCB$Q_NODE(R11),R6     ; first node spec
              0244 C9     D5    06B8   1725          TSTL    NWA$Q_ACS(R9)           ; branch if there is no defualt
                   2F     13    06BC   1726          BEQL    30$                     ;  access control string to apply
          54    44 AB40   7D    06BE   1727          MOVQ    FSCB$Q_NODE1(R11)[R0],R4         ; desc of last node
                54   02   A2    06C3   1728          SUBW2   #2,R4
                                06C6   1729
                                06C6   1730 ;
                                06C6   1731 ; Apply default access control string to nth node spec overriding the one
                                06C6   1732 ; (if any) associated with it.
                                06C6   1733 ;
                                06C6   1734 ; Note: <R4,R5> => last (nth) node spec (not including double colon delimiter)
                                06C6   1735 ;       <R6,R7> => 2nd to nth node spec string (including '::'s)
                                06C6   1736 ;
                                06C6   1737
             51    54     3C    06C6   1738          MOVZWL  R4,R1
             51    55     C0    06C9   1739          ADDL2   R5,R1                            ; R1 => end of last node
          07 54    12     E1    06CC   1740          BBC     #FSCB$V_ACS,R4,20$               ; branch if there is no access
                                06D0   1741                                                   ;  control string to override
          65 54    22     3A    06D0   1742          LOCC    #^A/"/,R4,(R5)                   ; <R0,R1> => ACS
             56    50     C2    06D4   1743          SUBL2   R0,R6                            ; Reduce size of string by size
                                06D7   1744                                                   ;  of ACS being removed
          56    0244 C9   A0    06D7   1745 20$:     ADDW2   NWA$Q_ACS(R9),R6                 ; increase size of string by
                                06DC   1746                                                   ;  size of ACS to be added
                0244 C9   28    06DC   1747          MOVC3   NWA$Q_ACS(R9),-                  ; insert default access control
          61    0248 D9         06E0   1748                  @NWA$Q_ACS+4(R9),(R1)            ;  string into last node spec
          63    3A3A 8F   B0    06E4   1749          MOVW    #^A/::7,(R3)                     ; add delimiter to string
                0244 C9   D4    06E9   1750          CLRL    NWA$Q_ACS(R9)                    ; zero ACS size/flag so we
                                06ED   1751                                                   ;  don't do it again
             56    56     3C    06ED   1752 30$:     MOVZWL  R6,R6                            ; clear and flags
        025C C9    56     C0    06F0   1753          ADDL2   R6,NWA$Q_INODE(R9)               ; update intermediate node spec
        0260 C9    56     C2    06F5   1754          SUBL2   R6,NWA$Q_INODE+4(R9)             ;  descriptor
   0080 8F    025C C9     B1    06FA   1755          CMPW    NWA$Q_INODE(R9),#NWA$C_INODESIZ  ; make sure that this node spec
                                0701   1756                                                   ;  will not overflow buffer
                   9F     1A    0701   1757          BGTRU   ERRNOD1                          ; branch on error
   0260 D9    67   56     28    0703   1758          MOVC3   R6,(R7),@NWA$Q_INODE+4(R9)       ; prefix node spec to the others
                          05    0709   1759          RSB                                      ; try another translation
```

```
                      070A  ;
                      070A  ; The last attempted logical node name translation failed--we have a real
                      070A  ; node name. Construct the primary node spec from this node name and any
                      070A  ; default access control string that may be in effect. Finally, update the
                      070A  ; node name element buffer with the primary and any intermediate node specs
                      070A  ; found so far.
                      070A  ;
                      070A  ; Note: <R4,R5> describes a node spec without '::', but possibly including a
                      070A  ; leading noise "_" character.
                      070A  ;
                      070A
                      070A  TRANSLATE_FAIL:
      07   54   B1    070A           CMPW     R4,#FWA$C_MAXNODNAM+1     ; check node name string size
           96   1A    070D           BGTRU    ERRNOD                    ; branch on error
      57   55   D0    070F           MOVL     R5,R7                     ; store address of string
      53   54   3C    0712           MOVZWL   R4,R3                     ; get address past string
    0244   C9   28    0715           MOVC3    NWA$Q_ACS(R9),-           ; append access control string (if any)
6543  0248 D9         0719                    @NWA$Q_ACS+4(R9),(R5)[R3]        ;  and double colon to construct
  83  3A3A 8F   B0    071E           MOVW     #^A/::7,(R3)+             ;  primary node spec string
  50   53   57  C3    0723           SUBL3    R7,R3,R0                  ; get new length (saving R6 flags)
      56   50   B0    0727           MOVW     R0,R6                     ; now <R6,R7> => node spec string
                      072A
                      072A  ;
                      072A  ; Validate and copy the primary (left-most) node spec to the node name element
                      072A  ; buffer. If node name has been seen during a previous pass (FNA or DNA), then
                      072A  ; this node spec will not be copied.
                      072A  ;
                      072A
      50   19   9A    072A           MOVZBL   #FWA$V_NODE,R0
      51   34   3C    072D           MOVZWL   #<FWA$C_MAXNODNAM+NWA$C_MAXACS+2>,R1
  55  00D8 CA   7E    0730           MOVAQ    FWA$Q_NODE(R10),R5
         064D   30    0735           BSBW     CHECK_FIELD
                      0738
                      0738  ;
                      0738  ; If no copy occurred, mark this to signify that intermediate and secondary
                      0738  ; node specs also should be ignored for this pass (DNA or RLF).
                      0738  ;
                      0738
        06 50   E8    0738           BLBS     R0,10$                    ; branch if primary node spec
                      073B                                              ;  was copied to element buffer
                      073B           SSB      #FWA$V_NOCOPY,(R10)       ; flag that the primary node spec
           3B   11    073F           BRB      50$                       ;  was not copied to element buffer
    00D8 CA     7D    0741  10$:     MOVQ     FWA$Q_NODE(R10),-         ; save descriptor of primary
    01B4 CA           0745                    FWA$Q_NODE1(R10)          ;  node spec string
           56   B4    0748           CLRW     R6                        ; clear size field
 01B4 CA   56   C8    074A           BISL2    R6,FWA$Q_NODE1(R10)       ; set flags in the descriptor
                      074F
                      074F  ;
                      074F  ; Append intermediate node specs (if any) that were generated from logical
                      074F  ; node name translation to the node name element buffer.
                      074F  ;
                      074F  ; First call scan_string to reparse the nodes, no need to check fo syntax
                      074F  ; and what ever since we built the names ourselves
                      074F  ;
                      074F
  56  025C C9   3C    074F           MOVZWL   NWA$Q_INODE(R9),R6        ; get descriptor of intermediate
           26   13    0754           BEQL     50$                       ;  node spec list, if null then exit
         57  0260 C9   D0  0756  1818          MOVL    NWA$Q_INODE+4(R9),R7      ; get address <R6,R7>
         5B  0104 CB   9E  075B  1819          MOVAB   FSCB$C_BLN(R11),R11       ; use secondary FSCB
                F89D'  30  0760  1820          BSBW    RM$SCAN_STRING            ; rescan the string to isolate
                           0763  1821                                            ;  intermediate node specs
                   5C  D4  0763  1822          CLRL    AP                        ; start loop
          01 AB    5C  91  0765  1823 30$:     CMPB    AP,FSCB$B_NODES(R11)      ; are we done
                   0C  1E  0769  1824          BGEQU   40$                       ; yes
         56   44 AB4C  7D  076B  1825          MOVQ    FSCB$Q_NODE1(R11)[AP],R6        ; get next node
                 00B9  30  0770  1826          BSBW    APPEND_NODESPEC           ; append next intermediate node spec
                           0773  1827                                            ;  to node name element buffer
                   5C  D6  0773  1828          INCL    AP                        ; index to the next nodespec
                   EE  11  0775  1829          BRB     30$                       ; loop
         5B  FEFC CB   9E  0777  1830 40$:     MOVAB   -FSCB$C_BLN(R11),R11      ; restore FSCB
                           077C  1831
                           077C  1832 ;
                           077C  1833 ; Logical node name translation performed on the left-most node spec of
                           077C  1834 ; the user supplied file specification string is complete. Now process any
                           077C  1835 ; other node specs and/or a quoted string that may follow.
                           077C  1836 ;
                           077C  1837
                   5C  D4  077C  1838 50$:     CLRL    AP                        ; index of node to use in
                           077E  1839                                            ;  secondary_node
```

```
                       077E  1841 ;
                       077E  1842 ; Check to see if the user specified any secondary node specs
                       077E  1843 ;
                       077E  1844
                       077E  1845 SECONDARY_NODE:
              5C   D6  077E  1846         INCL    AP                              ; get next node
      01 AB   5C   91  0780  1847         CMPB    AP,FSCB$B_NODES(R11)            ; any nodes left?
              07   1F  0784  1848         BLSSU   10$                             ; branch if yes
              59 8ED0  0786  1849         POPL    R9                              ; restore ifb
                       0789  1850         RMSSUC                                  ; exit wit success
                   05  078C  1851         RSB
                       078D  1852
                       078D  1853 ;
                       078D  1854 ; The user has supplied a secondary node spec string. Append it to the
                       078D  1855 ; node name element buffer, but do not perform logical node name translation
                       078D  1856 ; unless the node spec string has its password masked out.
                       078D  1857 ;
                       078D  1858
    56   44 AB4C   7D  078D  1859 10$:    MOVQ    FSCB$Q_NODE1(R11)[AP],R6       ; get descriptor of new node
      61 56   15   E1  0792  1860         BBC     #FSCB$V_PWD,R6,30$      ; branch if password is not masked out
                       0796  1861                                         ;  of access control string of node spec
       67  5F 8F   91  0796  1862         CMPB    #^A/_/,(R7)             ; remove leading underscore character
              04   12  079A  1863         BNEQ    20$                     ;  (if present) before attempting
              56   B7  079C  1864         DECW    R6                      ;  translation of special logical node
              57   D6  079E  1865         INCL    R7                      ;  name string with password masked out
                       07A0  1866
                       07A0  1867 ;
                       07A0  1868 ; Attempt one level of logical node name translation to obtain the original
                       07A0  1869 ; access control string because the current node spec string has its password
                       07A0  1870 ; masked out.
                       07A0  1871 ; Use of $TRNLOG here is okay as no extended logical name features are needed.
                       07A0  1872 ;
                       07A0  1873
                       07A0  1874         ASSUME  TRNLOG$_ACMODE  EQ      <TRNLOG$_DSBMSK-4>
                       07A0  1875         ASSUME  TRNLOG$_TABLE   EQ      <TRNLOG$_ACMODE-4>
                       07A0  1876         ASSUME  TRNLOG$_RSLBUF  EQ      <TRNLOG$_TABLE-4>
                       07A0  1877         ASSUME  TRNLOG$_RSLLEN  EQ      <TRNLOG$_RSLBUF-4>
                       07A0  1878         ASSUME  TRNLOG$_LOGNAM  EQ      <TRNLOG$_RSLLEN-4>
                       07A0  1879         ASSUME  TRNLOG$_NARGS   EQ      6
                       07A0  1880
         56   02   A2  07A0  1881 20$:    SUBW2   #2,R6                   ; remove trailing '::'
   023C C9    56   7D  07A3  1882         MOVQ    R6,NWA$Q_LOGNAME(R9)    ; store descriptor of node spec string
                       07A8  1883                                         ;  to translate
              03   DD  07A8  1884         PUSHL   #3                      ; only look in process table
              00   DD  07AA  1885         PUSHL   #0                      ; do not return mode of equiv string
              00   DD  07AC  1886         PUSHL   #0                      ; do not return table of equiv string
         024C C9   7F  07AE  1887         PUSHAQ  NWA$Q_XLTBUF1(R9)       ; specify address of descriptor of
                       07B2  1888                                         ;  buffer to receive equivalence string
         0230 C9   DF  07B2  1889         PUSHAL  NWA$L_XLTSIZ(R9)        ; specify address of word to receive
                       07B6  1890                                         ;  equivalence string size
         023C C9   7F  07B6  1891         PUSHAQ  NWA$Q_LOGNAME(R9)       ; specify address of descriptor pointing
                       07BA  1892                                         ;  to logical name candidate
                       07BA  1893         $TSTPT  NTXLATLOG               ;
00000000'9F   06   FB  07C0  1894         CALLS   #6,@#SYS$TRNLOG         ; Translate the logical name
         01   50   B1  07C7  1895         CMPW    R0,S^#SS$_NORMAL        ; Was translation successful?
              35   12  07CA  1896         BNEQ    ERRNOD2                 ; Branch if not
                       07CC  1897
   56  0230 C9   3C  07CC  1898          MOVZWL  NWA$L_XLTSIZ(R9),R6     ; descriptor of equivalence
   57  0250 C9   D0  07D1  1899          MOVL    NWA$Q_XLTBUF1+4(R9),R7  ;  string in <R6,R7>
                     07D6  1900
                     07D6  1901 ;
                     07D6  1902 ; Process the equivalence string.
                     07D6  1903 ;
                     07D6  1904 ; Note that a call to UPCASE_COMPRESS is not required.
                     07D6  1905 ;
                     07D6  1906
   5B  0104 CB   9E  07D6  1907          MOVAB   FSCB$K_BLN(R11),R11     ; use temporary FSCB
             56  DD  07DB  1908          PUSHL   R6                      ; save string length
          F820'  30  07DD  1909          BSBW    RM$SCAN_STRING          ; parse eq. string
          56   8ED0  07E0  1910          POPL    R6                      ; restore input length
     56   04 AB  B1  07E3  1911          CMPW    FSCB$Q_FILESPEC(R11),R6 ; was the full spec parsed ok
             18  12  07E7  1912          BNEQ    ERRNOD2                 ; no, exit
                     07E9  1913
                     07E9  1914          ASSUME  FSCB$B_FLDFLAGS EQ      0
                     07E9  1915
        6B   01  91  07E9  1916          CMPB    #FSCB$M_NODE,(R11)      ; better be just nodes
             13  12  07EC  1917          BNEQ    ERRNOD2                 ; no, exit
     56   0C AB  7D  07EE  1918          MOVQ    FSCB$Q_NODE(R11),R6     ; get descriptor of node spec
   5B  FEFC CB   9E  07F2  1919          MOVAB   -FSCB$K_BLN(R11),R11    ; restore FSCB
                     07F7  1920
                     07F7  1921 ;
                     07F7  1922 ; Validate and append secondary node spec to node name element buffer.
                     07F7  1923 ;
                     07F7  1924
             0B  10  07F7  1925 30$:     BSBB    PSEUDO_CHKFLD           ; validate node spec syntax
             31  10  07F9  1926          BSBB    APPEND_NODESPEC         ; append next secondary node spec
                     07FB  1927                                          ;  to node name element buffer
           FF80  31  07FB  1928          BRW     SECONDARY_NODE          ; examine next field
                     07FE  1929
                     07FE  1930 ;
                     07FE  1931 ; NOTE: We cannot return directly up to parse_string level because R9 was
                     07FE  1932 ;       pushed onto the stack in check_node so remove return pc
                     07FE  1933 ;       at errnod3 which will set the error code and return to parse_string
                     07FE  1934 ;
                     07FE  1935
                     07FE  1936 ERRNOD3:                                ;
          50   8ED0  07FE  1937          POPL    R0                      ; remove return PC
                     0801  1938 ERRNOD2:                                ;
           FEA1  31  0801  1939          BRW     ERRNOD                  ; Branch aid
```

```
                        0804   1941 ;
                        0804   1942 ; This routine determines whether a field is a possible node spec string
                        0804   1943 ; that may contain a logical node name and an access control string.
                        0804   1944 ;
                        0804   1945 ; Inputs:
                        0804   1946 ;
                        0804   1947 ;          <R6,R7> Descriptor of field
                        0804   1948 ;
                        0804   1949 ; Outputs:
                        0804   1950 ;
                        0804   1951 ;          R0-R2   Destroyed
                        0804   1952 ;
                        0804   1953
                        0804   1954 PSEUDO_CHKFLD:
52    56    02    A3    0804   1955          SUBW3   #2,R6,R2                  ; get string size w/o '::'
   0C 56    12    E1    0808   1956          BBC     #FSCB$V_ACS,R6,10$        ; branch if no ACS present
67    52    22    3A    080C   1957          LOCC    #^A\"\,R2,(R7)            ; on return <R0,R1> => ACS
      2C    50    B1    0810   1958          CMPW    R0,#NWA$C_MAXACS          ; check size of ACS
            09    1A    0813   1959          BGTRU   20$                       ; branch if too big
      52    50    A2    0815   1960          SUBW2   R0,R2                     ; compute size of logical node name
      0F    52    B1    0818   1961 10$:     CMPW    R2,#FWA$C_MAXLNDNAM       ; check size of logical node name
            E1    1A    081B   1962          BGTRU   ERRNOD3                   ; branch if too big
                  05    081D   1963          RSB                               ; exit
                        081E   1964
                        081E   1965 ;
                        081E   1966 ; NOTE: We cannot return directly up to parse_string level because R9 was
                        081E   1967 ;          pushed onto the stack in check_node so remove return pc, restore R9,
                        081E   1968 ;          set the error code and exit
                        081E   1969 ;
                        081E   1970
            50 8ED0     081E   1971 20$:     POPL    R0                        ; remove return PC
            59 8ED0     0821   1972          POPL    R9                        ; restore ifb ptr
                        0824   1973          RMSERR  ACS                       ; declare error in access cntrl string
         FD11     31    0829   1974          BRW     PRS_ERROR                 ; and get out
                        082C   1975
```

```
                              082C       ;
                              082C       ; This routine appends the string described by <R6,R7> to the node name element
                              082C       ; buffer described by FWA$Q_NODE, i.e., an intermediate or secondary node spec
                              082C       ; is appended to the node specs already in the node name element buffer.
                              082C       ; Also, a descriptor pointing to the appended string is saved in FWA.
                              082C       ;
                              082C       ; Inputs:
                              082C       ;
                              082C       ;       <R6,R7> Descriptor of field
                              082C       ;
                              082C       ; Outputs:
                              082C       ;
                              082C       ;       R0-R5 are destroyed.
                              082C       ;
                              082C       ;
                              082C       APPEND_NODESPEC:
 56    FFFF0000 8F      CA    082C               BICL2   #^XFFFF0000,R6            ; clr flags in 2nd word of descriptor
          2F 6A  01     E0    0833               BBS     #FWA$V_NOCOPY,(R10),10$   ; if primary node spec was discarded
                              0837                                                 ;  this pass, then discard this one too
             2F  AA     96    0837               INCB    FWA$B_SUBNODCNT(R10)      ; increment secondary node spec counter
      50     2F  AA     9A    083A               MOVZBL  FWA$B_SUBNODCNT(R10),R0   ; copy the counter
          07     50     D1    083E               CMPL    R0,#FWA$C_MAXSUBNOD       ; branch if there are too many
                 BB     1A    0841               BGTRU   ERRNOD3                   ;  secondary node specs
     52    00D8  CA     9E    0843               MOVAB   FWA$Q_NODE(R10),R2        ; get address of node descriptor
     53      62  82     C1    0848               ADDL3   (R2)+,(R2),R3             ; compute address of next available
                              084C                                                 ;  byte of node name buffer
 51     01B4 CA40       7E    084C               MOVAQ   FWA$Q_NODE1(R10)[R0],R1   ; get address of next node spec desc
             81  56     D0    0852               MOVL    R6,(R1)+                  ; save descriptor of node spec
             61  53     D0    0855               MOVL    R3,(R1)                   ;  being appended
             72  56     C0    0858               ADDL2   R6,-(R2)                  ; adjust size in node name descriptor
   007F  8F  62         B1    085B               CMPW    (R2),#FWA$C_MAXNODLST     ; make sure intermediate node spec
                              0860                                                 ;  string will fit in node name buffer
                 9C     1A    0860               BGTRU   ERRNOD3                   ; branch on error
     63      67  56     28    0862               MOVC3   R6,(R7),(R3)              ; append next node spec string
                              0866                                                 ;  to node name element buffer
                        05    0866       10$:    RSB                               ; exit
                              0867
```

```
                                0867  2015 ;
                                0867  2016 ; We have a file name
                                0867  2017 ;
                                0867  2018
                                0867  2019 CHECK_NAME:
                                0867  2020
                                0867  2021 ;
                                0867  2022 ; Check to see if it is a quoted string
                                0867  2023 ;
                                0867  2024
                                0867  2025         ASSUME  FWA$C_MAXQUOTED GE FWA$C_MAXNAME
                                0867  2026
        06 56    13     E1      0867  2027         BBC     #FSCB$V_QUOTED,R6,10$    ; is it a quoted name string?
                                086B  2028
        51    FF 8F     9A      086B  2029         MOVZBL  #FWA$C_MAXQUOTED,R1      ; for quoted strings use the max quoted size
                 03     11      086F  2030         BRB     20$                      ;
                                0871  2031
           51    27     9A      0871  2032 10$:    MOVZBL  #FWA$C_MAXNAME,R1        ; check non-quoted length
           50    0D     9A      0874  2033 20$:    MOVZBL  #FWA$V_NAME,R0           ; set code for check_field
     55  0170    CA     7E      0877  2034         MOVAQ   FWA$Q_NAME(R10),R5       ; output descriptor
             0506       30      087C  2035         BSBW    CHECK_FIELD              ; copy string
        08 6A    04     E1      087F  2036         BBC     #FWA$V_FNA_PASS,(R10),50$; branch if not FNA pass
        04 6A    0D     E1      0883  2037         BBC     #FWA$V_NAME,(R10),50$    ; branch if name not seen
                                0887  2038         SSB     #FWA$V_EXP_NAME,(R10)    ; set explicit name flag
           12 50        E9      088B  2039 50$:    BLBC    R0,90$                   ; if we didn't use this string, go on
        0E 56    13     E1      088E  2040         BBC     #FSCB$V_QUOTED,R6,90$    ; if string not quoted, continue
                                0892  2041
                                0892  2042 ;
                                0892  2043 ; We just used a quoted string for a name string - it now resides in
                                0892  2044 ; FWA$Q_NAME, quotes and all.
                                0892  2045 ;
                                0892  2046
                                0892  2047         SSB     #FWA$V_QUOTED,(R10)      ; flag quoted name in use
                                0896  2048
                                0896  2049 ;
                                0896  2050 ; We now will flag that we've seen a type, since having a quoted
                                0896  2051 ; string "implies" a type (that's why TYPE and not EXP_TYPE is set).
                                0896  2052 ; This keeps the type from showing up later by defaulting and relating
                                0896  2053 ; and such.  Unfortunately, we can't use the current status of the
                                0896  2054 ; FWA$V_TYPE bit to tell us whether we've already seen a type, as at the
                                0896  2055 ; end of the FNA scan EXP_TYPE if set is used to set TYPE.  EXP_TYPE is
                                0896  2056 ; set even if there was only a . with no "real" type characters (to
                                0896  2057 ; allow null types of course, otherwise they would fill in from the
                                0896  2058 ; defaults...).  So we have to check at this point for the length
                                0896  2059 ; of the FWA$Q_TYPE descriptor.
                                0896  2060 ;
                                0896  2061
                                0896  2062         SSB     #FWA$V_TYPE,(R10)
            0178 CA     D5      089A  2063         TSTL    FWA$Q_TYPE(R10)          ; is there any type?
                 04     12      089E  2064         BNEQ    FNMERR                   ; if neq yes, oops, error
                                08A0  2065
                                08A0  2066 90$:    RMSSUC
                 05             08A3  2067         RSB
                                08A4  2068
                                08A4  2069 ;
                                08A4  2070 ; return file name error here quick and easy
                                08A4  2071 ;
                 08A4  2072
      50     D4  08A4  2073 FNMERR: CLRL      R0
             05  08A6  2074         RSB
                 08A7  2075
                 08A7  2076
```

```
                      08A7        .SBTTL  PARSE_DIR, Parse a Directory String
                      08A7  ;
                      08A7  ; We have a left bracket (square or angle) indicating a directory spec.
                      08A7  ;
                      08A7  ; It may be of the [grp,mbr] format, [directory-name] format,
                      08A7  ; [directory-name.directory-name2...] format, or [.directory-name...]
                      08A7  ; format.
                      08A7  ;
                      08A7  ; Determine the format and copy the various parts to the directory buffers.
                      08A7  ;
                      08A7
                      08A7          ASSUME  FWA$C_MAXDIRLEN EQ      255     ; for overflow checking
                      08A7
                      08A7  PARSE_DIR:
            18    E0  08A7          BBS     #FSCB$V_CONCEAL,-               ; cant be concealed directory
      31 24 AB        08A9                  FSCB$Q_DIRECTORY(R11),ERRDIR1   ;  type
   08    03 AB    91  08AC          CMPB    FSCB$B_DIRS(R11),#FSCB$C_MAXDIR ; are thier to many directories?
            2B    1A  08B0          BGTRU   ERRDIR1                         ; yes, bad!
   28 BB    02    81  08B2          ADDB3   #2,@FSCB$Q_DIRECTORY+4(R11),-   ; determine whether to use ']'
         0A AA        08B6                  FWA$B_DIRTERM(R10)              ;  or '>' as teminator, taking
                      08B8                                                  ;  advantage of the fact that
                      08B8                                                  ;  the ASCII code for each of
                      08B8                                                  ;  the right brackets is 2 more
                      08B8                                                  ;  than the code for the
                      08B8                                                  ;  correspoinding left bracket
                      08B8
                      08B8  ;
                      08B8  ;   If this is the related file pass and we are doing an output file parse
                      08B8  ;   then if the directory in the current expanded filename is a
                      08B8  ;   wildcard - replace it with the first wild field in the directory
                      08B8  ;   of the related filename (i.e. sticky directories)
                      08B8  ;
                      08B8
       6A   03    E1  08B8          BBC     #FWA$V_RLF_PASS,(R10),- ; If this isn't a related file pass
            12        08BB                  10$                     ;  don't make directories sticky
       68   3D    E1  08BC          BBC     #FAB$V_OFP+FOP,(R8),-   ; If this isn't output file parse
            0E        08BF                  10$                     ;  don't make directories sticky
          0559    30  08C0          BSBW    STICKY_DIR              ; go check for sticky directories
         17 50    E9  08C3          BLBC    R0,ERRDIR1              ; if any errors, go report them
   0A AA    53    91  08C6          CMPB    R3,FWA$B_DIRTERM(R10)   ; have we passed the whole directory ?
            65    12  08CA          BNEQU   SKPDIR                  ; no - go eat the rest
            60    11  08CC          BRB     DIRXIT1                 ; only use sticky fields
                      08CE
   5F 6A    0E    E0  08CE  10$:    BBS     #FWA$V_DIR,(R10),SKPDIR ; Branch if directory already seen
 56   00C4  CB    7D  08D2          MOVQ    FSCB$Q_DIRECTORY1(R11),R6; get the first spec
   05 56    16    E0  08D7          BBS     #FSCB$V_GRPMBR,R6,-     ; is it [group,member] format?
                      08DB                  GRPMBR                  ;  or normal
            59    11  08DB          BRB     NAMED_DIR
                      08DD
                      08DD  ERRDIR1:
          00AB    31  08DD          BRW     ERRDIR                  ; Error in directory format
                      08E0
```

```
                              08E0  2131 ;
                              08E0  2132 ; We have a '[group,member]' directory format
                              08E0  2133 ;
                              08E0  2134
                              08E0  2135 GRPMBR:
                              08E0  2136
                              08E0  2137 ;
                              08E0  2138 ; check to make sure they are correct specs.
                              08E0  2139 ;
                              08E0  2140 ;   1.  3 characters or less
                              08E0  2141 ;   2.  less then or equal to 377 octal
                              08E0  2142 ;
                              08E0  2143
              03     56  B1   08E0  2144            CMPW    R6,#3                       ; 3 characters?
                     05  12   08E3  2145            BNEQ    10$                         ; no need to check unless 3
              33     67  91   08E5  2146            CMPB    (R7),#^A/3/                 ; is first char > 3?
                     F3  1A   08E8  2147            BGTRU   ERRDIR1                     ; if so it's an error
              50     20  9A   08EA  2148 10$:       MOVZBL  #FWA$V_DIR1,R0
              51     03  9A   08ED  2149            MOVZBL  #3,R1
      55    0130 CA  7E       08F0  2150            MOVAQ   FWA$Q_DIR1(R10),R5
                   048D  30   08F5  2151            BSBW    CHECK_FIELD
                              08F8  2152
                              08F8  2153 ;
                              08F8  2154 ; First part ok - check second part.
                              08F8  2155 ;
                              08F8  2156
      56    00CC CB  7D       08F8  2157            MOVQ    FSCB$Q_DIRECTORY2(R11),R6
              03     56  B1   08FD  2158            CMPW    R6,#3                       ; 3 characters?
                     05  12   0900  2159            BNEQ    20$                         ; no need to check unless 3
              33     67  91   0902  2160            CMPB    (R7),#^A/3/                 ; is first char > 3?
                     D6  1A   0905  2161            BGTRU   ERRDIR1                     ; if so it's an error
              50     21  9A   0907  2162 20$:       MOVZBL  #FWA$V_DIR2,R0
              51     03  9A   090A  2163            MOVZBL  #3,R1
      55    0138 CA  7E       090D  2164            MOVAQ   FWA$Q_DIR2(R10),R5
                   0470  30   0912  2165            BSBW    CHECK_FIELD
                              0915  2166
                              0915  2167            SSB     #FWA$V_GRPMBR,(R10)         ; flag group member format in fwa
                              0919  2168            SSB     #FSCB$V_GRPMBR,-            ;       ; and descriptor
                              0919  2169                    FWA$Q_DIR1(R10)
            0130 CA  81       091F  2170            ADDB3   FWA$Q_DIR1(R10),-           ; compute the directory length of the
            0138 CA           0923  2171                    FWA$Q_DIR2(R10),-           ;  UIC format directory by adding the
              2E AA           0926  2172                    FWA$B_DIRLEN(R10)           ;  lengths of the two separate parts
                              0928  2173                                                ;  together
                     B3  1F   0928  2174            BCS     ERRDIR1                     ; if overflow the byte
                              092A  2175            SSB     #FWA$V_DIR_LVLS,(R10)       ; treat UIC dirs as having 2 levels
                              092E  2176
                              092E  2177 DIRXIT1:
                   00C6  31   092E  2178            BRW     DIRXIT                      ; All set
                              0931  2179
                              0931  2180 ;
                              0931  2181 ; We must skip over directory spec.
                              0931  2182 ;
                              0931  2183
                              0931  2184            ASSUME  FWA$V_DUPOK      EQ      0
                              0931  2185
                  FA 6A  E8   0931  2186 SKPDIR:    BLBS    (R10),DIRXIT1               ; branch if duplicates allowed
                     55  11   0934  2187            BRB     ERRDIR
```

```
                          0936  2189 ;
                          0936  2190 ;        We have a normal format directory name.
                          0936  2191 ;        Check for '[]' or '[.name]' or '[-.name]'
                          0936  2192 ;        indicating explicit default directory.
                          0936  2193 ;
                          0936  2194
                          0936  2195 NAMED_DIR:
         54   01     CE   0936  2196         MNEGL   #1,R4                           ; set up for normal case
      04 56   17     E0   0939  2197         BBS     #FSCB$V_MINUS,R6,10$            ; relative if leading minus
      0C 56   14     E1   093D  2198         BBC     #FSCB$V_NULL,R6,30$             ; branch if not null
           00C9   30      0941  2199 10$:    BSBW    DEFAULT_DIR                     ; get default directory
           01 50     E8   0944  2200         BLBS    R0,20$                          ; ok, continue
                 05       0947  2201         RSB
                          0948  2202
              1D     EF   0948  2203 20$:    EXTZV   #FWA$V_DIR_LVLS,-               ; get the # of directory levels
      54   6A  03         094A  2204                 #FWA$S_DIR_LVLS,(R10),R4        ;  could be <>0 from default_dir
              5C     D4   094D  2205 30$:    CLRL    AP                              ; AP = dirs used from FSCB
                          094F  2206
                          094F  2207 ;
                          094F  2208 ; Subtract any minus signs from the directory levels
                          094F  2209 ;
                          094F  2210
      46 56   17     E1   094F  2211 LOOP:   BBC     #FSCB$V_MINUS,R6,COPY_DIR       ; are there any minus signs?
                          0953  2212
                          0953  2213 ;
                          0953  2214 ; The length of the directory string (R6) is the number of minus
                          0953  2215 ; signs present. NOTE: You cannot go past MFD but can go past UFD
                          0953  2216 ;
                          0953  2217
         51   56     3C   0953  2218         MOVZWL  R6,R1                           ; get number of '-'s
              54     D5   0956  2219 5$:     TSTL    R4                              ; can we subtract a directory
              1A     14   0958  2220         BGTR    10$                             ; ok, if >0
              2F     19   095A  2221         BLSS    ERRDIR                          ; can't go negative
                          095C  2222
                          095C  2223 ;
                          095C  2224 ; R4 = 0, we are at the top, check for MFD
                          095C  2225 ;
                          095C  2226
      06    0130 CA  B1   095C  2227         CMPW    FWA$Q_DIR1(R10),#6              ; MFD is six characters
              11     12   0961  2228         BNEQ    10$                             ; if not right then ok
      50    0134 CA  D0   0963  2229         MOVL    FWA$Q_DIR1+4(R10),R0            ; get address of name
         91'AF  80   D1   0968  2230         CMPL    (R0)+,B^MFD_ASCII               ; check if it's MFD
              06     12   096C  2231         BNEQ    10$                             ; neq, then it can't be
         91'AF  60   B1   096E  2232         CMPW    (R0),B^MFD_ASCII                ; last part
              17     13   0972  2233         BEQL    ERRDIR                          ; can't do that
      50   54   20   C1   0974  2234 10$:    ADDL3   #FWA$V_DIR1,R4,R0               ; get the flag offset
                          0978  2235         CSB     R0,(R10)                        ; clear the flag
   50   0130 CA44    7E   097C  2236         MOVAQ   FWA$Q_DIR1(R10)[R4],R0          ; get addr of descriptor
              60     D4   0982  2237         CLRL    (R0)                            ;  and clear it
              54     D7   0984  2238         DECL    R4                              ; remove this directory
           CD 51     F5   0986  2239         SOBGTR  R1,5$                           ; are there any more?
              34     11   0989  2240         BRB     GET_NEXT                        ; no, get next directory
                          098B  2241
                          098B  2242 ;
                          098B  2243 ; Handle directory error
                          098B  2244 ;
                          098B  2245
```

```
                           098B  ERRDIR: RMSERR  DIR                     ; show error
                       05  0990          RSB                             ;  and get out
                           0991
                           0991  ;
                           0991  ; ASCII name for the MFD and some trailing blanks
                           0991  ;
                           0991  MFD_ASCII:
  20 20 30 30 30 30 30 30  0991          .ASCII  ^000000  ^
                           0999
                           0999  ;
                           0999  ; Set up directory for check_field
                           0999  ;
                           0999
                           0999  COPY_DIR:
            07    54   D1  0999          CMPL    R4,#FWA$C_MAXSUBDIR             ; check to see if we are ok
                  ED   13  099C          BEQL    ERRDIR                          ; opps
         1D 56    14   E0  099E          BBS     #FSCB$V_NULL,R6,GET_NEXT                  ; null directory spec?
                  54   D6  09A2          INCL    R4                              ; ready to process it
      50    54    20   C1  09A4          ADDL3   #FWA$V_DIR1,R4,R0               ; find the flag offset
            51    27   9A  09A8          MOVZBL  #FWA$C_MAXNAME,R1               ; length
      55  0130  CA44   7E  09AB          MOVAQ   FWA$Q_DIR1(R10)[R4],R5          ; descriptor for this directory
                03D1   30  09B1          BSBW    CHECK_FIELD                     ; copy the directory
         2E AA    56   80  09B4          ADDB2   R6,FWA$B_DIRLEN(R10)            ; accumulate directory spec length
                  D1   1F  09B8          BCS     ERRDIR                          ; if overflow the byte
               2E AA   96  09BA          INCB    FWA$B_DIRLEN(R10)               ; count the terminator too
                  CC   1F  09BD          BCS     ERRDIR                          ; if overflow the byte
                           09BF
                           09BF  ;
                           09BF  ; Set up for next directory field
                           09BF  ;
                           09BF
                           09BF  GET_NEXT:
         18 56    10   E1  09BF          BBC     #FSCB$V_ELIPS,R6,10$            ; was there an elips?
                  54   D5  09C3          TSTL    R4                              ; < 0 if past UFD
                  02   18  09C5          BGEQ    5$                              ; ok, skip
                  32   10  09C7          BSBB    INSERT_MFD                      ; insert mfd in directories
      50  0130  CA44   7E  09C9  5$:     MOVAQ   FWA$Q_DIR1(R10)[R4],R0          ; get addr of descriptor
                           09CF          SSB     #FSCB$V_ELIPS,(R0)              ; set flag in high word
                           09D3          SSB     #FWA$V_WILDCARD,(R10)           ; set over all wild card
                           09D7          SSB     #FWA$V_WILD_DIR,(R10)           ; set directory wild card
                  5C   D6  09DB  10$:    INCL    AP                              ; inc. use count
         03 AB    5C   91  09DD          CMPB    AP,FSCB$B_DIRS(R11)             ; used them all?
                  09   13  09E1          BEQL    20$                             ; yes, finish up
      56  00C4  CB4C   7D  09E3          MOVQ    FSCB$Q_DIRECTORY1(R11)[AP],R6   ; get next directory
                FF63   31  09E9          BRW     LOOP                            ; go process it
                           09EC
                           09EC  ;
                           09EC  ; Directory parse completed - if directory specification contains no
                           09EC  ; levels (as a result of minus signs), then insert 000000 as directory name
                           09EC  ;
                           09EC
                  54   D5  09EC  20$:    TSTL    R4                              ; < 0 if past UFD
                  02   18  09EE          BGEQ    30$                             ; ok, skip
                  09   10  09F0          BSBB    INSERT_MFD                      ; insert mfd
            1D    54   F0  09F2  30$:    INSV    R4,#FWA$V_DIR_LVLS,-            ; stuff the final result
                  6A   03  09F5                  #FWA$S_DIR_LVLS,(R10)
                           09F7
```

```
                     09F7  2303 ;
                     09F7  2304 ; Directory processed ok, go process next field
                     09F7  2305 ;
                     09F7  2306
                     09F7  2307 DIRXIT: RMSSUC
                 05  09FA  2308         RSB
                     09FB  2309
                     09FB  2310
                     09FB  2311 ;
                     09FB  2312 ; Insert the name "000000" into the directory spec.
                     09FB  2313 ;
                     09FB  2314
                     09FB  2315 INSERT_MFD:
                     09FB  2316         SSB     #FWA$V_DFLT_MFD,(R10)           ; flag MFD was inserted
0130 CA   06     B0  09FF  2317         MOVW    #6,FWA$Q_DIR1(R10)              ; insert length of "000000"
                     0A04  2318                                                 ;  NOTE: Do not disturb flags
     8A AF       7D  0A04  2319         MOVQ    B^MFD_ASCII,-                   ; move the string "000000"
   0134 DA           0A07  2320                 @FWA$Q_DIR1+4(R10)
        54       D4  0A0A  2321         CLRL    R4                              ; say just one level
                 05  0A0C  2322         RSB
```

```
                          0A0D  ;
                          0A0D  ; We have a directory spec of the form [] or [.name], or [-.name]
                          0A0D  ; implying a sub-directory of the current default directory.
                          0A0D  ; It is valid only if default directory is not of the [group,member] format.
                          0A0D  ; Call PARSE_STRING recursively to apply default directory.
                          0A0D  ;
                          0A0D  ;
                          0A0D
                          0A0D  DEFAULT_DIR:
                          0A0D
                          0A0D  ;
                          0A0D  ; If this is a network filespec, then the directory string should be sent
                          0A0D  ; to the remote FAL as entered, that is, without resolving hyphens or a
                          0A0D  ; null root directory. So if a node name is present, say UFD seen and
                          0A0D  ; make the corresponding FWA descriptor point to a null root directory
                          0A0D  ; string or one consisting of one or more hyphens.
                          0A0D  ;
                          0A0D
      13 6A    19    E1   0A0D            BBC     #FWA$V_NODE,(R10),10$      ; branch if node not seen
    0130 CA    56    D0   0A11            MOVL    R6,FWA$Q_DIR1(R10)         ; store string size in DIR1 descriptor
      67       56    28   0A16            MOVC    R6,(R7),-                  ; copy the string
      0134 DA             0A19                    @FWA$Q_DIR1+4(R10)
                          0A1C            CSB     #FSCB$V_MINUS,R6           ; clear minus bit in mask to avoid
                          0A20                                               ;  having it resolved in this routine
      54 6A    20    E3   0A20            BBCS    #FWA$V_DIR1,(R10),20$      ; say UFD seen and branch
                          0A24
                          0A24  ;
                          0A24  ; And its context (logname flag and directory terminator).
                          0A24  ;
                          0A24
       00FC 8F       BB   0A24  10$:      PUSHR   #^M<R2,R3,R4,R5,R6,R7>     ; save current registers
     7E    06 AA     90   0A28            MOVB    FWA$B_LNFLGS(R10),-(SP)    ;
     7E    0A AA     90   0A2C            MOVB    FWA$B_DIRTERM(R10),-(SP)   ;
           06 AA     94   0A30            CLRB    FWA$B_LNFLGS(R10)          ; say no logical name seen
                          0A33            CSB     #FWA$V_DIR,(R10)           ; clear directory seen flag
                          0A37
                          0A37  ;
                          0A37  ; Call PARSE_DIR recursively
                          0A37  ;
                          0A37
                          0A37            ASSUME  FSCB$B_FLDFLAGS EQ      0
                          0A37
     5B   0104 CB    9E   0A37            MOVAB   FSCB$K_BLN(R11),R11        ; use temporary FSCB
                6B   94   0A3C            CLRB    (R11)                      ; clear flags
 57   00000000'9F    9E   0A3E            MOVAB   @#PIO$GT_DDSTRING,R7       ; make descriptor of default directory
          56    87   9A   0A45            MOVZBL  (R7)+,R6                   ;  string (stored as counted string)
                56   DD   0A48            PUSHL   R6                         ; save input length
             F5B3'   30   0A4A            BSBW    RM$SCAN_STRING             ; parse default directory
             56 8ED0      0A4D            POPL    R6                         ; restore input length
                50   D4   0A50            CLRL    R0                         ; assume failure
     56    04 AB     B1   0A52            CMPW    FSCB$Q_FILESPEC(R11),R6    ; was the full spec parsed ok
                0C   12   0A56            BNEQ    15$                        ; no, exit
        6B    08     91   0A58            CMPB    #FSCB$M_DIRECTORY,(R11)    ; should only have a directory
                07   12   0A5B            BNEQ    15$                        ; incorrect default string if so.
     56    24 AB     7D   0A5D            MOVQ    FSCB$Q_DIRECTORY(R11),R6          ; get string
              FE43   30   0A61            BSBW    PARSE_DIR                  ; parse it
                          0A64
```

```
                               0A64   2381 ;
                               0A64   2382 ; Restore previous context.
                               0A64   2383 ;
                               0A64   2384
        5B   FEFC CB     9E    0A64   2385 15$:      MOVAB    -FSCB$K_BLN(R11),R11      ; restore old FSCB
          0A AA   8E     90    0A69   2386           MOVB     (SP)+,FWA$B_DIRTERM(R10);
          06 AA   8E     90    0A6D   2387           MOVB     (SP)+,FWA$B_LNFLGS(R10) ;
             00FC 8F     BA    0A71   2388           POPR     #^M<R2,R3,R4,R5,R6,R7>    ; restore registers
                15 50    E9    0A75   2389           BLBC     R0,90$                    ; branch on failure
                               0A78   2390
                               0A78   2391 ;
                               0A78   2392 ; If done with directory spec, then exit immediately else check for
                               0A78   2393 ; compatible directory formats.
                               0A78   2394 ;
                               0A78   2395
        06 56     17     E0    0A78   2396 20$:      BBS      #FSCB$V_MINUS,R6,30$      ; If leading minus,
        03 AB     01     91    0A7C   2397           CMPB     #1,FSCB$B_DIRS(R11)       ;   or if not done
                  04     13    0A80   2398           BEQL     40$                       ;   then handle relative filespec
                               0A82   2399
                               0A82   2400 ;
                               0A82   2401 ; Do not allow concatenation of uic format directories.
                               0A82   2402 ;
                               0A82   2403
        04 6A     1B     E0    0A82   2404 30$:      BBS      #FWA$V_GRPMBR,(R10),50$ ; format ok?
           50     01     9A    0A86   2405 40$:      MOVZBL   #1,R0                     ; success
                         05    0A89   2406           RSB
                               0A8A   2407
             FEFE        31    0A8A   2408 50$:      BRW      ERRDIR                    ; no, get error then exit
                               0A8D   2409
                               0A8D   2410 ;
                               0A8D   2411 ; The default directory string is messed up. This should
                               0A8D   2412 ; never happen, but it's not too serious. Try to continue.
                               0A8D   2413 ;
                               0A8D   2414
                               0A8D   2415 90$:      RMSERR   BUG_DDI                   ; return error
                         05    0A92   2416           RSB
```

```
                                  0A93          .SBTTL TRANSLATE, Translate Logical Name
                                  0A93
                                  0A93  ;
                                  0A93  ; Logical name translation ...
                                  0A93  ;
                                  0A93  ; This subroutine attempts to translate the string described by the last
                                  0A93  ; FWA$Q_LOGNAM string.  If this is NOT a search list logical name the current
                                  0A93  ; SLB, R6, will be the 'fake' SLB in the fwa itself.  If it is a known search
                                  0A93  ; list logical name use the index (which could have been updated) in the
                                  0A93  ; appropiate SLB.
                                  0A93  ;
                                  0A93  ; If we are successful, check for an 'escape' equivalence string indicating a
                                  0A93  ; process-permanent file in which case the escape string is stored in the
                                  0A93  ; FWA and the parse is terminated.
                                  0A93  ;
                                  0A93  ; If not an escape string, the original string is returned to the user
                                  0A93  ; and a success return is made.
                                  0A93  ;
                                  0A93  ; If not an escape string or underline, the equivalence string replaces the
                                  0A93  ; parse input string and a branch is made to PARSE_STRING to merge in the
                                  0A93  ; filename elements of the equivalence string. If the translate was
                                  0A93  ; unsuccessful, a failure return is made to the caller with the equivalence
                                  0A93  ; string (identical to the input name minus the initial underscore if any)
                                  0A93  ; described by <R6,R7>.
                                  0A93  ;
                                  0A93  ; This returns only if the string was not used either the translation failed
                                  0A93  ; or we did not want to use it.  Errors are returned one level up.  If the
                                  0A93  ; name was translated and used we branch to parse_string and act like we are
                                  0A93  ; continuing to parse the same string (same FSCB).
                                  0A93  ;
                                  0A93  ; If it returns:
                                  0A93  ;
                                  0A93  ;       If R0 = 1 then  <R6,R7> => orginal input string (Concealed device)
                                  0A93  ;
                                  0A93  ;       If R0 = 0 then  <R6,R7> => orginal input string without leading
                                  0A93  ;                                  underscore, if any (No translation)
                                  0A93  ;
                                  0A93
                                  0A93
                                  0A93  LNM_ATTR:
                         02000000 0A93          .LONG   LNM$M_CASE_BLIND        ; Caseless LNM translation
56 45 44 5F 45 4C 49 46 24 4D 4E 4C 0A97  TABNAM: .ASCII  /LNM$FILE_DEV/          ; RMS logical table name string
                         0000000C 0AA3  TABNAM_SIZE = .-TABNAM
                                  0AA3
                                  0AA3  ;
                                  0AA3  ; Try to translate logical name
                                  0AA3  ;
                                  0AA3
                                  0AA3  TRANSLATE:
                          5C 8ED0 0AA3          POPL    AP      ; remove (save) return PC, we are now at the PARSE_STRING
                                  0AA6                          ; level, so PRS_EXIT and PRS_ERROR are legal branches
                                  0AA6
                                  0AA6  ;
                                  0AA6  ; Search the SLBs (if any) to see if one matches the level of recursion
                                  0AA6  ; we are at, if so use the index from it.  If non found use the fake one.
                                  0AA6  ;
                                  0AA6
```

```
         56  00AC CA   D0  0AA6  2475            MOVL    FWA$L_SLB_PTR(R10),R6              ; get the current slb
                  15   13  0AAB  2476            BEQL    10$                                ; branch if none
         50       56   D0  0AAD  2477            MOVL    R6,R0                              ; save for end test
                           0AB0  2478
                           0AB0  2479    ;
                           0AB0  2480    ; Check to see if we are at a level where search list are present, if so
                           0AB0  2481    ; then use the SLB for this level.  If not continue if with the fake SLB.
                           0AB0  2482    ;
                           0AB0  2483
   00C3 CA     0B A6   91  0AB0  2484    5$:     CMPB    SLB$B_LEVEL(R6),FWA$B_LEVEL(R10); check the level
                  0F   13  0AB6  2485            BEQL    15$                                ; if same then use it
                  08   14  0AB8  2486            BGTR    10$                                ; if past this level stop
         56       66   D0  0ABA  2487            MOVL    SLB$L_FLINK(R6),R6                 ; get the next one
         56       50   D1  0ABD  2488            CMPL    R0,R6                              ; at the end
                  EE   12  0AC0  2489            BNEQ    5$                                 ; no, continue
         56  00B8 CA   DE  0AC2  2490    10$:    MOVAL   FWA$T_SLB(R10),R6                  ; use the fake one
                           0AC7  2491
                           0AC7  2492    ;
                           0AC7  2493    ; Check to see if we have done the max number of translations
                           0AC7  2494    ;
                           0AC7  2495
      0B A6       01   81  0AC7  2496    15$:    ADDB3   #1,SLB$B_LEVEL(R6),-               ; bump the count and move it
           00C3 CA         0ACB  2497                    FWA$B_LEVEL(R10)                   ;  into the fwa
      0A   00C3 CA     91  0ACE  2498            CMPB    FWA$B_LEVEL(R10),#LNM$C_MAXDEPTH ; max trans it 10
                  64   14  0AD3  2499            BGTR    ERRLNE                             ; were ok, continue
                           0AD5  2500
                           0AD5  2501    ;
                           0AD5  2502    ; R6 - SLB with index to translate
                           0AD5  2503    ;
                           0AD5  2504
      14 A6       09   E1  0AD5  2505    TRAN:   BBC     #LNM$V_TERMINAL,SLB$L_ATTR(R6),-; is this name a terminal?
                  03       0AD9  2506                    10$                                ; if not continue
                0105   31  0ADA  2507            BRW     NOTRAN                             ; else don't bother to translate
                           0ADD  2508
                           0ADD  2509    ;
                           0ADD  2510    ; Create a item list to pass to $trnlnm,
                           0ADD  2511    ;
                           0ADD  2512
               0C A6   DE  0ADD  2513    10$:    MOVAL   SLB$L_INDEX(R6),-                  ; get the index buffer addr
               60 AA       0AE0  2514                    FWA$T_ITM_INDEX+4(R10)
               14 A6   DE  0AE2  2515            MOVAL   SLB$L_ATTR(R6),-                   ; get the attr buffer addr
               6C AA       0AE5  2516                    FWA$T_ITM_ATTR+4(R10)
                           0AE7  2517
                           0AE7  2518            ASSUME  FWA$L_XLTBUFF2  EQ      FWA$L_XLTBUFF1+4
                           0AE7  2519
      50     009C CA   98  0AE7  2520            CVTBL   FWA$B_BUFFLG(R10),R0               ; which buffer should we use?
                           0AEC  2521                                                       ; 0 => buff2; -1 => buff1
    009C CA       50   92  0AEC  2522            MCOMB   R0,FWA$B_BUFFLG(R10)               ; switch buffers for next time
  55   00A4 CA40       D0  0AF1  2523            MOVL    FWA$L_XLTBUFF2(R10)[R0],R5         ; get output buffer addr
      78 AA       55   D0  0AF7  2524            MOVL    R5,FWA$T_ITM_STRING+4(R10)         ;
               10 A6   DE  0AFB  2525            MOVAL   SLB$L_MAX_INDEX(R6),-              ; get the max_index buffer addr
             0084 CA       0AFE  2526                    FWA$T_ITM_MAX_INDEX+4(R10)         ;
               93 AF   9F  0B01  2527            PUSHAB  TABNAM                             ; make descriptor of tablename
                  0C   DD  0B04  2528            PUSHL   #TABNAM_SIZE
      51          5E   D0  0B06  2529            MOVL    SP,R1
                           0B09  2530
                           0B09  2531            $TSTPT  XLATLOG
                           0B0F
                           0B0F   TRNLNM: $TRNLNM_S -
                           0B0F                           TABNAM=(R1),-
                           0B0F                           ATTR=W^LNM_ATTR,-
                           0B0F                           LOGNAM=FWA$Q_LOGNAM(R10),-
                           0B0F                           ITMLST=FWA$T_ITMLST(R10),-
                           0B0F                           ACMODE=FWA$B_XLTMODE(R10)
                           0B27
        5E     08     C0   0B27                 ADDL2   #8,SP                            ; discard tablename descriptor
54    009E     CA     3C   0B2A                 MOVZWL  FWA$W_XLTSIZ(R10),R4             ; <R4,R5> => translation if any
           0F  50     E8   0B2F                 BLBS    R0,NORMAL
01BC 8F        50     B1   0B32                 CMPW    R0,#SS$_NOLOGNAM
           1C         13   0B37                 BEQL    NOLOGNAM
                           0B39
                           0B39   ;
                           0B39   ; Either the system service failed or the maximum number of translations
                           0B39   ; has been exceeded or the logical name translation yielded an escape
                           0B39   ; string that was either the wrong length, of the wrong mode, or not
                           0B39   ; the only field.
                           0B39   ;
                           0B39
                           0B39   ERRLNE: RMSERR  LNE                              ; show error
         F9FC         31   0B3E           BRW     PRS_ERROR                        ; and get out
                           0B41
                           0B41   ;
                           0B41   ; Something worked.  Lets see if it really did anything usefull
                           0B41   ;
                           0B41
       10 A6          D5   0B41   NORMAL: TSTL    SLB$L_MAX_INDEX(R6)              ; was there really a translation
          0F          19   0B44           BLSS    NOLOGNAM                         ; no
  14 A6   0A          E0   0B46           BBS     #LNM$V_EXISTS,SLB$L_ATTR(R6),-   ; was there a translation for
          34               0B4A                   HAVEONE                          ;  this index?
       0C A6          D6   0B4B           INCL    SLB$L_INDEX(R6)                  ; bump index
       0C A6          D1   0B4E           CMPL    SLB$L_INDEX(R6),-                ; have we gone past the max
       10 A6               0B51                   SLB$L_MAX_INDEX(R6)              ;  index for this name?
          BA          15   0B53           BLEQ    TRNLNM                           ; no, try again, NOTE: the item
                           0B55                                                    ;  list is still valid
                           0B55
                           0B55   ;
                           0B55   ; We did not get a translation of any kind, remove the current SLB (if any)
                           0B55   ; and continue on
                           0B55   ;
                           0B55
                           0B55   NOLOGNAM:
       0C A6          D4   0B55           CLRL    SLB$L_INDEX(R6)                  ; clear the index
       0B A6          97   0B58           DECB    SLB$B_LEVEL(R6)                  ; and lower the level (in fwa)
  0A A6   00          E1   0B5B           BBC     #SLB$V_REALSLB,SLB$B_FLAGS(R6),-; do we realy need to deallocate
          1C               0B5F                   20$
                           0B60
                           0B60   ;
                           0B60   ; Deallocate SLB
                           0B60   ;
                           0B60
     54   66          0F   0B60           REMQUE  (R6),R4                          ; remove the SLB from the list
          08          1C   0B63           BVC     10$                              ; last one?
50    00A8 CA         D0   0B65           MOVL    FWA$L_SLBH_PTR(R10),R0           ; get SLBH address
       0C A0          D4   0B6A           CLRL    SLBH$L_SLB_QUE(R0)               ; clear the pointer to the list
```

```
          5B   DD  0B6D  2589 10$:    PUSHL   R11                                 ; save FSCB
       5B 54 AA   D0  0B6F  2590            MOVL    FWA$L_IMPURE_AREA(R10),R11          ; restore impure address
    00000000'EF   16  0B73  2591            JSB     RM$RETBLK1                          ; deallocate the space
        5B 8ED0  0B79  2592            POPL    R11                                 ; restore FSCB
          0063   31  0B7C  2593 20$:    BRW     NOTRAN                              ; return failure
                     0B7F  2594
                     0B7F  2595 ;
                     0B7F  2596 ; We translated a logical name.  Check to see if the name we translated was
                     0B7F  2597 ; a search list.
                     0B7F  2598 ;
                     0B7F  2599
                     0B7F  2600 HAVEONE:
          10 A6   D5  0B7F  2601            TSTL    SLB$L_MAX_INDEX(R6)                 ; was this a search list
             0F   15  0B82  2602            BLEQ    20$                                 ; no
                     0B84  2603
                     0B84  2604 ;
                     0B84  2605 ; We have incountered a search list logical name, it could be one we already
                     0B84  2606 ; know about or a new one
                     0B84  2607 ;
                     0B84  2608
                     0B84  2609            SSB     #FWA$V_SLPRESENT,(R10)              ; flag that we have one
       0A A6   00   E0  0B88  2610            BBS     #SLB$V_REALSLB,SLB$B_FLAGS(R6),-; did we already know that?
               06       0B8C  2611                    20$
                     0B8D  2612
                     0B8D  2613 ;
                     0B8D  2614 ; Allocate a SLB place it in the queue and copy the logical name form
                     0B8D  2615 ; the fake SLB to it, R6 will point to the new SLB
                     0B8D  2616 ;
                     0B8D  2617
          0111   30  0B8D  2618            BSBW    ALLOCATE_SLB                        ; get an SLB
          1A 50   E9  0B90  2619            BLBC    R0,60$                              ; if there is one
                     0B93  2620
                     0B93  2621 ;
                     0B93  2622 ; Check to see if we got an escape string indicating a process permanent file.
                     0B93  2623 ;
                     0B93  2624
        1B    65   91  0B93  2625 20$:    CMPB    (R5),#ESCAPE                        ; do we have a PPF?
              18   13  0B96  2626            BEQL    PPERMFILE                           ; yes, process it
                     0B98  2627
                     0B98  2628 ;
                     0B98  2629 ; If the equivalence string has a concealed device then ignore the
                     0B98  2630 ; translation and preserve the logical name as the device name.
                     0B98  2631 ;
                     0B98  2632
       14 A6   08   E0  0B98  2633 30$:    BBS     #LNM$V_CONCEALED,SLB$L_ATTR(R6),-; concealed device?
               5A       0B9C  2634                    PARSE_CONCEAL
                     0B9D  2635
                     0B9D  2636 ;
                     0B9D  2637 ; Parse the equivalence string as the filename string.
                     0B9D  2638 ;
                     0B9D  2639
        56    54   7D  0B9D  2640            MOVQ    R4,R6                               ; get into right registers
            0172   30  0BA0  2641            BSBW    UPCASE_COMPRESS                     ; perform upcasing and string
                     0BA3  2642                                                        ;  compression in place
       03 6A   04   E1  0BA3  2643            BBC     #FWA$V_FNA_PASS,(R10),50$           ; leave dupok if not fna pass
          6A   01   8A  0BA7  2644            BICB2   #FWA$M_DUPOK,(R10)                  ; logical names must provide
                     0BAA  2645                                                        ;  no duplicate fields
```

```
                    F826   31  0BAA  2646 50$:      BRW      PARSE_STRING                        ; continue processing
                               0BAD  2647
                    F98D   31  0BAD  2648 60$:      BRW      PRS_ERROR                           ; quit if no room
```

```
                           0BB0  2650 ;
                           0BB0  2651 ; Ppermfile ...
                           0BB0  2652 ;
                           0BB0  2653 ; The equivalence string starts with an escape.
                           0BB0  2654 ; It looks like we have a process permanent file.
                           0BB0  2655 ;
                           0BB0  2656 ; Check that the equivalence string has at least 4 bytes.
                           0BB0  2657 ; Don't use it if it is an RLFi string.
                           0BB0  2658 ;
                           0BB0  2659
                           0BB0  2660 PPERMFILE:
        2B 6A    03    E0  0BB0  2661              BBS     #FWA$V_RLF_PASS,(R10),30$         ; ignore if processing RLFi
           04    54    D1  0BB4  2662              CMPL    R4,#4                             ; is escape string 4 bytes long?
                 23    1F  0BB7  2663              BLSSU   20$                               ; no - uninterpretable
        0C AA    65    D0  0BB9  2664              MOVL    (R5),FWA$L_ESCSTRING(R10)         ; store the escape string
                           0BBD  2665
                           0BBD  2666 ;
                           0BBD  2667 ; Use CHECK_FIELD to copy the process permanent logical name
                           0BBD  2668 ; to the device name field.
                           0BBD  2669 ;
                           0BBD  2670
     56    00D0 CA     7D  0BBD  2671              MOVQ    FWA$Q_LOGNAM(R10),R6              ; get descriptor
              50   0F  9A  0BC2  2672              MOVZBL  #FWA$V_DEVICE,R0
         51    FF 8F   9A  0BC5  2673              MOVZBL  #FWA$S_DEVICEBUF,R1
     55    00E0 CA     7E  0BC9  2674              MOVAQ   FWA$Q_DEVICE(R10),R5
                 01B4  30  0BCE  2675              BSBW    CHECK_FIELD
        04 6A    04    E1  0BD1  2676              BBC     #FWA$V_FNA_PASS,(R10),10$         ; branch if not processing FNA
                           0BD5  2677              SSB     #FWA$V_EXP_DEV,(R10)              ; set explicit device flag
              F95B     31  0BD9  2678 10$:         BRW     PRS_EXIT1                         ; all done, exit parse_string
                           0BDC  2679
              FF5A     31  0BDC  2680 20$:         BRW     ERRLNE                            ; error in translation
                           0BDF  2681
              F912     31  0BDF  2682 30$:         BRW     PRS_EXIT                          ; all done, ignore this string
```

```
                         0BE2  2684 ;
                         0BE2  2685 ; No translation occurred
                         0BE2  2686 ;
                         0BE2  2687
              50   D4    0BE2  2688 NOTRAN: CLRL    R0                              ; show no translation
                         0BE4  2689 ;;;; strip leading underscore if there is one
                         0BE4  2690 ;;;        movq    fwa$q_lognam(r10),r6            ; restore input string
                         0BE4  2691 ;;;        cmpb    (r7),#^a/_/                     ; leading underscore?
                         0BE4  2692 ;;;        bneq    extran                          ; no, exit
                         0BE4  2693 ;;;        decw    r6                              ; cut it off
                         0BE4  2694 ;;;        incl    r7                              ;
              0A   11    0BE4  2695         BRB     EXTRAN                          ; exit
                         0BE6  2696
                         0BE6  2697 ;
                         0BE6  2698 ; The translation is not to be used (Concealed device/directory)
                         0BE6  2699 ;
                         0BE6  2700
                         0BE6  2701 DONTUSE:
         50   01   9A    0BE6  2702         MOVZBL  #1,R0                           ; translation valid, but not to be used
       009C CA   92      0BE9  2703         MCOMB   FWA$B_BUFFLG(R10),-             ; flip the buffer flag back
       009C CA           0BED  2704                 FWA$B_BUFFLG(R10)
                         0BF0  2705
  56   00D0 CA   7D      0BF0  2706 EXTRAN: MOVQ    FWA$Q_LOGNAM(R10),R6            ; restore input logical name
              6C   17    0BF5  2707         JMP     (AP)                            ; effective RSB to caller
                         0BF7  2708
```

```
                   0BF7          .SBTTL  PARSE_CONCEAL, Parse Concealed Logical Name
                   0BF7
                   0BF7  ;
                   0BF7  ; PARSE_CONCEAL
                   0BF7  ;
                   0BF7  ;       Parse the concealed device string and rooted directory string, if any
                   0BF7  ;
                   0BF7
                   0BF7  PARSE_CONCEAL:
  45 6A   0F    E0 0BF7          BBS     #FWA$V_DEVICE,(R10),80$                ; Branch if device already seen
  41 6A   39    E2 0BFB          BBSS    #FWA$V_CONCEAL_DEV,(R10),80$           ; have we seen one before?
     56   54    7D 0BFF          MOVQ    R4,R6                                  ; save input string
        0110    30 0C02          BSBW    UPCASE_COMPRESS                        ; copy and compress string
5B     0104 CB  9E 0C05          MOVAB   FSCB$K_BLN(R11),R11                    ; use temporary FSCB
          56    DD 0C0A          PUSHL   R6                                     ; save size
       F3F1'    30 0C0C          BSBW    RM$SCAN_STRING                         ; parse the parts
          56  8ED0 0C0F          POPL    R6                                     ; restore input length
  56   04 AB    B1 0C12          CMPW    FSCB$Q_FILESPEC(R11),R6                ; was the full spec parsed ok
          2A    12 0C16          BNEQ    90$                                    ; no, exit
                   0C18
                   0C18          ASSUME  FSCB$B_FLDFLAGS EQ      0
                   0C18
                   0C18  ;
                   0C18  ; Make sure the concealed device name is of the form "DEVICE:" or
                   0C18  ; "DEVICE:[ROOTED-DIR.]" -- nothing more, nothing less.
                   0C18  ;
                   0C18
          06    8B 0C18          BICB3   #<FSCB$M_DEVICE!FSCB$M_ROOT>,-         ; check for illegal fields
     50   6B       0C1A                  (R11),R0                               ;
          24    12 0C1C          BNEQ    90$                                    ; br -- caught one!
  20 6B   01    E1 0C1E          BBC     #FSCB$V_DEVICE,(R11),90$               ; make sure it includes dev nam
                   0C22
                   0C22  ;
                   0C22  ; Copy the device descriptor and directory descriptor if any into the fwa
                   0C22  ;
                   0C22
                   0C22          ASSUME  FSCB$Q_ROOT8    EQ      FSCB$Q_ROOT1+56
                   0C22          ASSUME  FWA$Q_CDIR8     EQ      FWA$Q_CDIR1+56
                   0C22
  14 AB   01    A3 0C22          SUBW3   #1,FSCB$Q_DEVICE(R11),-                ; get device length and
     00E8 CA       0C26                  FWA$Q_CONCEAL_DEV(R10)                 ; remove trailing ':'
     00E8 CA    28 0C29          MOVC3   FWA$Q_CONCEAL_DEV(R10),-               ; copy string
       18 BB       0C2D                  @FSCB$Q_DEVICE+4(R11),-
     00EC DA       0C2F                  @FWA$Q_CONCEAL_DEV+4(R10)
  05 6B   02    E1 0C32          3BC     #FSCB$V_ROOT,(R11),10$                 ; is there a root directory
          12    10 0C36          BSBB    CHECK_ROOT                             ; copy the roots
       07 50    E9 0C38          BLBC    R0,90$                                 ; signal error
5B   FEFC CB    9E 0C3B  10$:    MOVAB   -FSCB$K_BLN(R11),R11                   ; restore first FSCB
          A4    11 0C40  80$:    BRB     DONTUSE                                ;  and ignore the translation
                   0C42
                   0C42  90$:    RMSERR  DEV                                    ; say it's a device error
        F8F3    31 0C47          BRW     PRS_ERROR
                   0C4A
```

```
                          0C4A  ;
                          0C4A  ; This routine copies the rooted directories from the FSCB into the FWA.
                          0C4A  ;
                          0C4A
                          0C4A  CHECK_ROOT:
              56     DD   0C4A          PUSHL   R6                              ; save r6
                          0C4C
                          0C4C  ;
                          0C4C  ; We must follow special rules if this is the filename from a related
                          0C4C  ; file block.  We want to include the rooted dir (e.g. [SOPHIE.CHOICE.])
                          0C4C  ; in basically the same instances that we would include a device.
                          0C4C  ;
                          0C4C
       0B 6A     03      E1   0C4C          BBC     #FWA$V_RLF_PASS,(R10),5$        ; br if not related filename
    6A    8040   8F      B3   0C50          BITW    #FWA$M_DEVICE!FWA$M_EXP_NODE,(R10) ;device or node seen already?
                 3C      12   0C55          BNEQ    20$                             ; br to ignore root if so
       38 68     3D      E0   0C57          BBS     #FAB$V_OFP+FOP,(R8),20$         ; ignore rootdir on output parse
                          0C5B
       3B 6A     3A      E2   0C5B  5$:     BBSS    #FWA$V_ROOT_DIR,(R10),40$       ; say root dir present
       56     02 AB      9A   0C5F          MOVZBL  FSCB$B_ROOTS(R11),R6            ; add number of concealed dirs
          08     56      91   0C63          CMPB    R6,#8                           ; better not be more then 8 dirs
                 35      14   0C66          BGTR    50$                             ; if not error
       53   0088 DB      9E   0C68          MOVAB   @FSCB$Q_ROOT1+4(R11),R3         ; get addr of start of string
 0B AA     FF A3 02      81   0C6D          ADDB3   #2,-1(R3),FWA$B_ROOTERM(R10)    ; stuff the terminator
                          0C73
                          0C73  ;
                          0C73  ; Can't be wild, have elips or minus signs
                          0C73  ; Count needs to be adjusted for zero-origin as well.
                          0C73  ;
                          0C73
      51    7C AB46      7E   0C73  10$:    MOVAQ   FSCB$Q_ROOT1-8(R11)[R6],R1      ; get addr of FSCB descriptors
    52    00E8 CA46      7E   0C78          MOVAQ   FWA$Q_CDIR1-8(R10)[R6],R2       ; get addr of fwa descriptors
  61    00830000 8F      D3   0C7E          BITL    #<FSCB$M_WILD!FSCB$M_ELIPS!FSCB$M_MINUS>,(R1)
                 16      12   0C85          BNEQ    50$                             ; if so error!
          62     61      B0   0C87          MOVW    (R1),(R2)                       ; copy length
 04 B2     04 B1 61      28   0C8A          MOVC3   (R1),@4(R1),@4(R2)              ; copy string
              E0 56      F5   0C90          SOBGTR  R6,10$                          ; loop for all descriptors
                          0C93  20$:    RMSSUC                                  ; ok
              56 8ED0         0C96  30$:    POPL    R6                              ; and r6
                 05           0C99          RSB                                     ; exit
                          0C9A
                          0C9A          ASSUME  FWA$V_DUPOK     EQ      0
                          0C9A
              F6 6A      E8   0C9A  40$:    BLBS    (R10),20$                       ; ok/ignore if duplicate
                 50      D4   0C9D  50$:    CLRL    R0                              ; error
                 F5      11   0C9F          BRB     30$                             ; exit
                          0CA1
                          0CA1
```

```
                       0CA1           .SBTTL  ALLOCATE_SLB, Allocate Search List Block
                       0CA1
                       0CA1  ;++
                       0CA1  ; ALLOCATE_SLB
                       0CA1  ;
                       0CA1  ;       Allocate a SLB to be put on the xx_SLB list.  The information
                       0CA1  ;       in the SLB is to be copied from the 'FAKE' SLB in the FWA
                       0CA1  ;       pointed to by R6.
                       0CA1  ;
                       0CA1  ;       On exit R6 will point to the new SLB.
                       0CA1  ;
                       0CA1  ;       Things to set set:
                       0CA1  ;
                       0CA1  ;               SLB$L_FLINK     = linked into slb list
                       0CA1  ;               SLB$L_BLINK     =         "
                       0CA1  ;               SLB$B_BLN       = SLB$C_BLN
                       0CA1  ;               SLB$B_FLAGS     = SLB$V_REALSLB
                       0CA1  ;
                       0CA1  ;
                       0CA1  ;       Things to be copied from 'fake' SLB:
                       0CA1  ;
                       0CA1  ;               SLB$B_LEVEL     = fake$b_level - 1
                       0CA1  ;               SLB$L_INDEX
                       0CA1  ;               SLB$L_MAX_INDEX
                       0CA1  ;               SLB$L_ATTR
                       0CA1  ;--
                       0CA1
                       0CA1  ;
                       0CA1  ; Allocate a SLB
                       0CA1  ;
                       0CA1
                       0CA1  ALLOCATE_SLB:
          7E   54   7D 0CA1          MOVQ    R4,-(SP)                        ; save R4,R5
          52   06   9A 0CA4          MOVZBL  #<SLB$C_BLN/4>,R2               ; get # of longwords
               5B   DD 0CA7          PUSHL   R11                             ; save FSCB
     5B   54   AA   D0 0CA9          MOVL    FWA$L_IMPURE_AREA(R10),R11      ; restore impure address
       00000000'EF  16 0CAD          JSB     RM$GETBLK1                      ; get the space
               5B 8ED0 0CB3          POPL    R11                             ; restore FSCB
          38   50   E9 0CB6          BLBC    R0,90$                          ; exit on error
                       0CB9
                       0CB9  ;
                       0CB9  ; We have a new SLB pointed to by R1, format it and add it to the END of
                       0CB9  ; the search list queue for this pass.
                       0CB9  ;
                       0CB9
                       0CB9          ASSUME  SLB$L_FLINK     EQ      0
                       0CB9          ASSUME  SLB$L_BLINK     EQ      SLB$L_FLINK+4
                       0CB9          ASSUME  SLB$B_BID       EQ      SLB$L_BLINK+4
                       0CB9          ASSUME  SLB$B_BLN       EQ      SLB$B_BID+1
                       0CB9          ASSUME  SLB$B_FLAGS     EQ      SLB$B_BLN+1
                       0CB9          ASSUME  SLB$B_LEVEL     EQ      SLB$B_FLAGS+1
                       0CB9          ASSUME  SLB$L_INDEX     EQ      SLB$B_LEVEL+1
                       0CB9          ASSUME  SLB$L_MAX_INDEX EQ      SLB$L_INDEX+4
                       0CB9          ASSUME  SLB$L_ATTR      EQ      SLB$L_MAX_INDEX+4
                       0CB9
          56   51   D0 0CB9          MOVL    R1,R6                           ; copy addr
          81   56   D0 0CBC          MOVL    R6,(R1)+                        ; set up forward and
```

```
         81    56   D0  0CBF           MOVL    R6,(R1)+                          ;  backward pointers
                        0CC2
 52    00A8    CA   D0  0CC2           MOVL    FWA$L_SLBH_PTR(R10),R2            ; get SLBH addr
   50    0C    A2   D0  0CC7           MOVL    SLBH$L_SLB_QUE(R2),R0             ; get first SLB
               06   13  0CCB           BEQL    10$                               ; there is none
    04 B0      66   0E  0CCD           INSQUE  (R6),@SLB$L_BLINK(R0)             ; insert at tail
               04   11  0CD1           BRB     20$                               ; continue
    0C A2      56   D0  0CD3      10$: MOVL    R6,SLBH$L_SLB_QUE(R2)             ; add only element
                        0CD7
         81    29   90  0CD7      20$: MOVB    #SLB$C_BID,(R1)+                  ; set block id
               51   D6  0CDA           INCL    R1                                ; skip BLN
         81    01   90  0CDC           MOVB    #SLB$M_REALSLB,(R1)+              ; set flag
                        0CDF
                        0CDF           ASSUME  SLB$B_LEVEL      EQ       <FWA$B_LEVEL-FWA$T_SLB>
                        0CDF
 50    00C3    CA   DE  0CDF           MOVAL   FWA$B_LEVEL(R10),R0               ; get 'fake' SLB addr at INDEX
 81    80      01   83  0CE4           SUBB3   #1,(R0)+,(R1)+                    ; copy level - 1
       81      80   7D  0CE8           MOVQ    (R0)+,(R1)+                       ; copy index and max_index
       81      80   D0  0CEB           MOVL    (R0)+,(R1)+                       ; copy attr
                        0CEE           RMSSUC
         54    8E   7D  0CF1      90$: MOVQ    (SP)+,R4                          ; restore R4,R5
                    05  0CF4           RSB
```

```
                          0CF5         .SBTTL  UPCASE_COMPRESS Upcase and compress user strings
                          0CF5  ;++
                          0CF5  ;
                          0CF5  ; Entry point for UPCASE_COMPRESS.
                          0CF5  ;
                          0CF5  ; This routine copies the source string to the destination buffer while
                          0CF5  ; upcasing lowercase characters and removing space, horizontal tab, and null
                          0CF5  ; characters, except that characters enclosed in quotes are moved unaltered.
                          0CF5  ;
                          0CF5  ; Inputs:
                          0CF5  ;
                          0CF5  ;      R5      Destination buffer descriptor
                          0CF5  ;      <R6,R7> Source string descriptor
                          0CF5  ;
                          0CF5  ; Outputs:
                          0CF5  ;
                          0CF5  ;      R5      Destroyed
                          0CF5  ;      <R6,R7> Destination string descriptor
                          0CF5  ;
                          0CF5  ;--
                          0CF5
                          0CF5  ;
                          0CF5  ;  Bit mask of escape characters when upcasing and compressing
                          0CF5  ;  Chars are quote (22), blank (20), tab (9) and null (0).
                          0CF5  ;
                          0CF5
           00000201      0CF5  ESCAPE_CHAR:    .LONG   ^B00000000000000000000001000000001  ; null and tab
           00000005      0CF9                  .LONG   ^B00000000000000000000000000000101  ; space and quote
           00000000      0CFD                  .LONG   ^B00000000000000000000000000000000
           00000000      0D01                  .LONG   ^B00000000000000000000000000000000
           00000000      0D05                  .LONG   ^B00000000000000000000000000000000
           00000000      0D09                  .LONG   ^B00000000000000000000000000000000
           00000000      0D0D                  .LONG   ^B00000000000000000000000000000000
           00000000      0D11                  .LONG   ^B00000000000000000000000000000000
                          0D15
                          0D15
                          0D15  UPCASE_COMPRESS:                                   ;
               56    D5   0D15          TSTL    R6                                 ; Branch if source string length is
               20    13   0D17          BEQL    50$                                ;  zero
          7E   54    7D   0D19          MOVQ    R4,-(SP)                           ; Save destination buffer address
                          0D1C                                                     ;  and free register
                          0D1C
                          0D1C  ;
                          0D1C  ; Examine each character in the source string.
                          0D1C  ; Take advantage of the fact that the ASCII values for space, horizontal tab,
                          0D1C  ; null, and quote are less than the ASCII values for lowercase characters.
                          0D1C  ;
                          0D1C
          54   87    9A   0D1C  10$:    MOVZBL  (R7)+,R4                           ; Get char
     16 D2 AF   54    E0   0D1F          BBS     R4,B^ESCAPE_CHAR,60$               ; A special one?
 85  00000000'GF44    90   0D24  20$:    MOVB    G^EXE$UPCASE_DAT[R4],-             ; Upcase it
                          0D2C                  (R5)+                              ; and append to destination
           ED 56     F5   0D2C  30$:    SOBGTR  R6,10$                             ; Check for end of source string
                          0D2F
                          0D2F  ;
                          0D2F  ; The source string is exhausted.
                          0D2F  ;
```

```
                            0D2F
                  54 8ED0   0D2F  40$:    POPL    R4                      ; Restore saved register
                  57 8ED0   0D32          POPL    R7                      ; Restore destination address
     56     55    57   C3   0D35          SUBL3   R7,R5,R6                ; Compute length of destination string
                       05   0D39  50$:    RSB                             ; Exit with <R6,R7> describing
                            0D3A                                          ;  upcased and compressed string
                            0D3A
                            0D3A  ;
                            0D3A  ; A punctuation mark was found, check for which one
                            0D3A  ; A space, tab, or null character has been found, simply discard it.
                            0D3A  ;
                            0D3A
            22    54   91   0D3A  60$:    CMPB    R4,#^A/"/               ; Check for quote
                  ED   12   0D3D          BNEQ    30$                     ; Discard character
                            0D3F
                            0D3F  ;
                            0D3F  ; A leading quote character has been found.
                            0D3F  ; Copy all characters unaltered until the matching quote is found, unless
                            0D3F  ; the leading quote was preceeded by two  colons which indicates that this is
                            0D3F  ; the start of a network quoted string (as opposed to an access control string)
                            0D3F  ;
                            0D3F
 7E    55     04  AE   C3   0D3F  70$:    SUBL3   4(SP),R5,-(SP)          ; Which char pos is this in?
            85    54   90   0D44          MOVB    R4,(R5)+                ; Store the quote
                  56   D7   0D47          DECL    R6                      ; and count it
                  06   14   0D49          BGTR    80$                     ; branch if not at end of string
       5E     04  AE   DE   0D4B          MOVAL   4(SP),SP                ; pop top element from stack
                  DE   11   0D4F          BRB     40$                     ; and branch to exit
                  8E   D7   0D51  80$:    DECL    (SP)+                   ; Branch if first or second character
                  08   15   0D53          BLEQ    90$                     ;  of string
 3A3A 8F      FE  A5   B1   0D55          CMPW    -2(R5),#^A\::\          ; Branch if previous two non-compressed
                  10   13   0D5B          BEQL    100$                    ;  character were both colons
                            0D5D
                            0D5D  ;
                            0D5D  ; This is where we skip all characters between quoted, and do not upcase or
                            0D5D  ; compress them.  This works for access control strings because they don't
                            0D5D  ; have imbedded quotes, and it works for ANSI-"a" filespecs (which may have
                            0D5D  ; imbedded quotes) because the imbedded quotes are doubled.  That causes them
                            0D5D  ; to look like lots of short quoted strings as far as this routine is concerned
                            0D5D  ; but nevertheless, this routine will still not bother the internal characters.
                            0D5D  ;
                            0D5D  ; Note that ANSI-"a" filespecs do want to be upcased -- this is done at the
                            0D5D  ; end of processing -- when we actually append the type and version fields
                            0D5D  ;
                            0D5D
            54    87   9A   0D5D  90$:    MOVZBL  (R7)+,R4                ; Get next char
            22    54   91   0D60          CMPB    R4,#^A\"\               ; Is character the matching quote?
                  BF   13   0D63          BEQL    20$                     ; Branch if yes to resume normal scan
            85    54   90   0D65          MOVB    R4,(R5)+                ; Append character to destination string
               F2 56   F5   0D68          SOBGTR  R6,90$                  ; End of source string?
                  C2   11   0D6B          BRB     40$                     ; Branch if yes
                            0D6D
                            0D6D  ;
                            0D6D  ; This is the start of a network quoted string. It may have embedded quote
                            0D6D  ; characters and it must be the last element of a file specification string.
                            0D6D  ; Therefore, copy all characters unaltered until end of string is reached,
                            0D6D  ; then work backwards to remove trailing spaces, tabs, and nulls.
```

```
                 0D6D   3007 ;
                 0D6D   3008
85   87   90     0D6D   3009 100$:    MOVB    (R7)+,(R5)+                   ; Append the rest of the source string
  FA 56   F5     0D70   3010          SOBGTR  R6,100$                       ;  to the destination string
                 0D73   3011
                 0D73   3012 ;
                 0D73   3013 ; The source string has been exhausted. Truncate string after last non
                 0D73   3014 ; space, tab, or null character.
                 0D73   3015 ;
                 0D73   3016
20   75   91     0D73   3017 110$:    CMPB    -(R5),#^A/ /                  ; Discard space
     FB   13     0D76   3018          BEQL    110$                          ;
09   65   91     0D78   3019          CMPB    (R5),#HOR_TAB                 ; Discard horizontal tab
     F6   13     0D7B   3020          BEQL    110$                          ;
     65   95     0D7D   3021          TSTB    (R5)                          ; Discard null
     F2   13     0D7F   3022          BEQL    110$                          ;
     55   D6     0D81   3023          INCL    R5                            ; Position pointer past new end of
     AA   11     0D83   3024          BRB     40$                           ;  destination string
```

```
0D85          .SBTTL  CHECK_FIELD - Check for Valid Field and Copy Routine
0D85
0D85   ;++
0D85   ;
0D85   ; Functional Description:
0D85   ;
0D85   ; This routine checks the field described by input descriptor for
0D85   ; validity based upon the field flags the high word of the descriptor and
0D85   ; and the desired checks specified by R0.
0D85   ;
0D85   ; Checks and operations performed:
0D85   ;
0D85   ;       1.  if the field is null, return.
0D85   ;       1a. check the size
0D85   ;       2.  if FWA$V_RLF_PASS set and the field is not
0D85   ;           selected by the type of parse (input or output, based
0D85   ;           upon the FOP bit OFP), return, thus ignoring the field.
0D85   ;       3.  if the filename element specified has already been seen, then
0D85   ;           return unless FWA$V_DUPOK is clear (duplicates
0D85   ;           not allowed) in which case generate an error.
0D85   ;       4.  flag the filename element present.
0D85   ;       5.  if the field is a wild card, flag that too,
0D85   ;           both for specific field and overall.
0D85   ;       6.  if this is a logical name save the descriptor only,
0D85   ;           otherwise copy the field to the element buffer setting the
0D85   ;           length in the descriptor.
0D85   ;       7.  return
0D85   ;       8.  set V_NOT_COPIED if duplicate field was ignored
0D85   ;           (i.e., not copied to element buffer).
0D85   ;
0D85   ; Calling Sequence:
0D85   ;
0D85   ;       BSBW    CHECK_FIELD
0D85   ;
0D85   ; Input Parameters:
0D85   ;
0D85   ;       R0      Flags
0D85   ;       R1      Maximun field size
0D85   ;       R6,R7   Descriptor of input field
0D85   ;       R5      Address of descriptor of output field
0D85   ;       R8      FAB address
0D85   ;       R10     FWA address
0D85   ;
0D85   ; Implicit Inputs:
0D85   ;
0D85   ;       The current state of the FWA, including
0D85   ;       FWA$B_FLDFLGS, FWA$V_RLF_PASS,  FWA$V_DUPOK, FWA$V_LOGNAME
0D85   ;
0D85   ; Outputs:
0D85   ;
0D85   ;       R0 -    1       - Ok, copied
0D85   ;               0       - Ok, not copied
0D85   ;
0D85   ;       R1-R3,R5        Destroyed
0D85   ;
0D85   ; Implicit Outputs:
0D85   ;
```

```
                     0D85  3083 ;           -FWA$B_FLDFLGS or FWA$V_LOGNAME or FWA$V_NODE set if field copied
                     0D85  3084 ;           -FWA$B_WILDFLGS and FWA$V_WILD set if field is wild
                     0D85  3085 ;           -FWA$B_DIRFLGS set if directory field (also FWA$B_DIRWCFLGS)
                     0D85  3086 ;           -the indicated FWA element buffer and its descriptor are
                     0D85  3087 ;            updated if field copied
                     0D85  3088 ;
                     0D85  3089 ; Completion Codes:
                     0D85  3090 ;
                     0D85  3091 ;           R0
                     0D85  3092 ;
                     0D85  3093 ; Side Effects:
                     0D85  3094 ;
                     0D85  3095 ;           None
                     0D85  3096 ;
                     0D85  3097 ;--
                     0D85  3098
                     0D85  3099 CHECK_FIELD:
      51    56   B1  0D85  3100         CMPW    R6,R1                           ; is the field size ok?
            23   1A  0D88  3101         BGTRU   10$                             ; nope
   39 6A    03   E1  0D8A  3102         BBC     #FWA$V_RLF_PASS,(R10),40$       ; Branch if not parse of
                     0D8E  3103                                                 ;  RLF string
                     0D8E  3104
                     0D8E  3105 ;
                     0D8E  3106 ; This is the parse of the related file string.
                     0D8E  3107 ; Look at the OFP FOP bit to determine whether this field is
                     0D8E  3108 ; to be utilized or not.
                     0D8E  3109 ;
                     0D8E  3110 ; The rules are:
                     0D8E  3111 ;
                     0D8E  3112 ;           If this is a parse of an input file all fields except version are sticky.
                     0D8E  3113 ;           including logicalnames  unless the field is null or
                     0D8E  3114 ;           unless a node (real or null) has been specified,
                     0D8E  3115 ;           then the node, device and directory fields are ignored.
                     0D8E  3116 ;
                     0D8E  3117 ;           If this is a parse of an output file spec, only the filename and type
                     0D8E  3118 ;           are sticky. (Sticky means they will be defaulted from the related file
                     0D8E  3119 ;           resultant filename string.)
                     0D8E  3120 ;
                     0D8E  3121 ;           Additionally, if the name, type, directory or
                     0D8E  3122 ;           version of an output file string contains a wild card, this implies
                     0D8E  3123 ;           an explicit default from the related filename string.
                     0D8E  3124 ;
                     0D8E  3125
                     0D8E  3126 ;
                     0D8E  3127 ; This is an output file parse.
                     0D8E  3128 ; Ignore node, device and logname fields.
                     0D8E  3129 ; For name and type fields, check for wild card and if so use the RLF field,
                     0D8E  3130 ; else just use if not already seen.
                     0D8E  3131 ; For dir and version fields, check for wild card and if so use the RLF field,
                     0D8E  3132 ; else ignore.
                     0D8E  3133 ;
                     0D8E  3134
                     0D8E  3135         ASSUME  FWA$V_WC_NAME   EQ      <FWA$V_NAME+8>
                     0D8E  3136         ASSUME  FWA$V_WC_TYPE   EQ      <FWA$V_TYPE+8>
                     0D8E  3137         ASSUME  FWA$V_WC_VER    EQ      <FWA$V_VERSION+8>
                     0D8E  3138
   1D 68    3D   E1  0D8E  3139         BBC     #FAB$V_OFP+FOP,(R8),20$ ; if not output file parse
```

```
              19   50   91  0D92  3140            CMPB    R0,#FWA$V_NODE           ; Is this the node?
                   70   13  0D95  3141            BEQL    CHKRTN                   ; ignore if so
              0F   50   91  0D97  3142            CMPB    R0,#FWA$V_DEVICE         ; Is this the device?
                   6B   13  0D9A  3143            BEQL    CHKRTN                   ; ignore if so
        4D 01 AA   50   E4  0D9C  3144            BBSC    R0,1(R10),80$            ; Use if wild and make non-wild
              0D   50   91  0DA1  3145            CMPB    R0,#FWA$V_NAME           ; is this a name?
                   25   13  0DA4  3146            BEQL    50$                      ; Go default the field if not seen
              0C   50   91  0DA6  3147            CMPB    R0,#FWA$V_TYPE           ; is this a type?
                   20   13  0DA9  3148            BEQL    50$                      ; Go default the field if not seen
                   5A   11  0DAB  3149            BRB     CHKRTN                   ; Ignore version and dir if not wild
                            0DAD  3150
                   67   11  0DAD  3151 10$:       BRB     CHKERR                   ; no, signal error
                            0DAF  3152
                            0DAF  3153 ;
                            0DAF  3154 ; This is an input file parse:
                            0DAF  3155 ; Ignore version field.
                            0DAF  3156 ; Use name, type and logname fields if not null.
                            0DAF  3157 ; Use node, device and directory fields if not null and node not specified.
                            0DAF  3158 ;
                            0DAF  3159
              0B   50   91  0DAF  3160 20$:       CMPB    R0,#FWA$V_VERSION        ; Is this the version?
                   53   13  0DB2  3161            BEQL    CHKRTN                   ; ignore if so
              30   50   91  0DB4  3162            CMPB    R0,#FWA$V_LOGNAME        ; is this a logical name?
                   0E   13  0DB7  3163            BEQL    40$                      ; yep
              0D   50   91  0DB9  3164            CMPB    R0,#FWA$V_NAME           ; is this a name?
                   09   13  0DBC  3165            BEQL    40$                      ; yep
              0C   50   91  0DBE  3166            CMPB    R0,#FWA$V_TYPE           ; is this a type?
                   04   13  0DC1  3167            BEQL    40$                      ; yep
           40 6A   06   E0  0DC3  3168            BBS     #FWA$V_EXP_NODE,(R10),CHKRTN    ; Ignore if node specified
                            0DC7  3169                                             ; Use field if not null
                            0DC7  3170
                            0DC7  3171 ;
                            0DC7  3172 ; The field is acceptable for use.   Use if not null and not duplicate.
                            0DC7  3173 ;
                            0DC7  3174
           3C 56   14   E0  0DC7  3175 40$:       BBS     #FSCB$V_NULL,R6,CHKRTN          ; branch if null field
           3B 6A   50   E2  0DCB  3176 50$:       BBSS    R0,(R10),CHKDUP                 ; branch if duplicate
                            0DCF  3177                                                    ; Else show we have it now ...
                            0DCF  3178 ;
                            0DCF  3179 ; Check for directory field and set the overall directory flag if so
                            0DCF  3180 ;
                04 AA   95  0DCF  3181            TSTB    FWA$B_DIRFLGS(R10)       ; Any directory bits set,
                   04   13  0DD2  3182            BEQL    60$                      ;  No - go away
                            0DD4  3183            SSB     #FWA$V_DIR,(R10)         ;  Yes - set the dir summary bit
                            0DD8  3184
                            0DD8  3185
                            0DD8  3186 ;
                            0DD8  3187 ; Check for wild card.
                            0DD8  3188 ;
                            0DD8  3189
           12 56   11   E1  0DD8  3190 60$:       BBC     #FSCB$V_WILD,R6,80$      ; Branch if field not a wild card
                            0DDC  3191            SSB     #FWA$V_WILDCARD,(R10)    ; Set overall flag
                            0DE0  3192
                            0DE0  3193            ASSUME  FWA$V_WC_VER    EQ      <FWA$V_VERSION+8>
                            0DE0  3194            ASSUME  FWA$V_WILD_UFD  EQ      <FWA$V_DIR1+8>
                            0DE0  3195
                            0DE0  3196            SSB     R0,1(R10)                ; Set specific wild card
```

```
                 0DE5  3197                                             ; At same bit position in next byte
      05 AA   95 0DE5  3198             TSTB    FWA$B_DIRWCFLGS(R10)    ; Any directory wild card bits set,
         04   13 0DE8  3199             BEQL    80$                     ;  No - go away
                 0DEA  3200             SSB     #FWA$V_WILD_DIR,(R10)   ;  Yes - set the wild dir summary bit
                 0DEE  3201
                 0DEE  3202     ;
                 0DEE  3203     ; If this is a logical name don't copy the string.
                 0DEE  3204     ;
                 0DEE  3205
      30   50   91 0DEE  3206 80$:     CMPB    R0,#FWA$V_LOGNAME       ; if this is a logical name then
           11   13 0DF1  3207          BEQL    CPYDSC                  ;  branch
           54   DD 0DF3  3208          PUSHL   R4
                 0DF5  3209
      85   56   3C 0DF5  3210          MOVZWL  R6,(R5)+                ; store length
00 B5 67   56   28 0DF8  3211          MOVC3   R6,(R7),@(R5)           ;  and copy the string
                 0DFD  3212
           54 8ED0 0DFD  3213          POPL    R4
      50   01   9A 0E00  3214 CHKSUC:  MOVZBL  #1,R0                   ; exit to caller with success
                05 0E03  3215          RSB
                 0E04  3216
                 0E04  3217     ;
                 0E04  3218     ; Don't copy of string since it's a logical name, just save descriptor.
                 0E04  3219     ;
                 0E04  3220
      65   56   7D 0E04  3221 CPYDSC:  MOVQ    R6,(R5)                 ; save descriptor
                 0E07  3222
                 0E07  3223     ;
                 0E07  3224     ; Return success and string_not_copied
                 0E07  3225     ;
                 0E07  3226
           50   D4 0E07  3227 CHKRTN:  CLRL    R0                      ; flag field not copied
                05 0E09  3228          RSB                             ; return
                 0E0A  3229
                 0E0A  3230     ;
                 0E0A  3231     ; This field has already been set.
                 0E0A  3232     ; Ignore it as long as FWA$V_DUPOK is set.
                 0E0A  3233     ;
                 0E0A  3234
                 0E0A  3235          ASSUME  FWA$V_DUPOK     EQ      0
                 0E0A  3236
        FA 6A   E8 0E0A  3237 CHKDUP:  BLBS    (R10),CHKRTN
                 0E0D  3238
                 0E0D  3239     ;
                 0E0D  3240     ; If this is duplicate device because of NAM block input, ignore
                 0E0D  3241     ; the duplicate.
                 0E0D  3242     ;
                 0E0D  3243
      0F   50   91 0E0D  3244          CMPB    R0,#FWA$V_DEVICE        ; Is this a device?
           04   12 0E10  3245          BNEQ    CHKERR                  ; Branch if not - solid error
      6A   05   E0 0E12  3246          BBS     #FWA$V_NAM_DVI,(R10),-  ; Branch if device came from DVI
                F1 0E15  3247                  CHKRTN                  ;  in NAM block, ignoring duplicate
                 0E16  3248     ;
                 0E16  3249     ; An error has been encountered.
                 0E16  3250     ;
                 0E16  3251
           50 8ED0 0E16  3252 CHKERR:  POPL    R0                      ; pop return pc
           50   D4 0E19  3253          CLRL    R0                      ; flag error
```

```
05  0E1B  3254          RSB                                   ; exit on level up
    0E1C  3255
```

```
0E1C              .SBTTL  STICKY_DIR, Process Sticky Directories
0E1C      ;++
0E1C      ;
0E1C      ; Functional Description:
0E1C      ;
0E1C      ;       This routine substitutes subdirectories from the RLF name block
0E1C      ;       for wild subdirectories in the expanded directory string.
0E1C      ;       The algorithm is:
0E1C      ;
0E1C      ;               if the output-file directory is in UIC format
0E1C      ;                   replace wild group part with related-file group part
0E1C      ;                   replace wild member part with related-file member part
0E1C      ;
0E1C      ;               if the output-file directory is wild (ie- [*] or [*...] ONLY)
0E1C      ;                   move the entire related-file directory specification
0E1C      ;                   if the related-file directory is in UIC format
0E1C      ;                       mark the output-file directory as in UIC format
0E1C      ;
0E1C      ;               if the output-file directory has a trailing ellipsis ("...")
0E1C      ;                   move the entire related-file subdirectory tree starting
0E1C      ;                       from the first wild related-file (sub)directory
0E1C      ;                   if there is no wild related-file (sub)directory, just
0E1C      ;                       remove the ellipsis
0E1C      ;
0E1C      ;               if the output-file directory contains trailing asterisks ("*")
0E1C      ;                   substitute for the trailing asterisks, related-file
0E1C      ;                       (sub)directories, starting with the first wild
0E1C      ;                       related-file (sub)directory
0E1C      ;
0E1C      ;       Additional Syntax Rules:
0E1C      ;
0E1C      ;       1. No substitution occurs when the output-file directory specification
0E1C      ;          is in UFD format and partially wild, but there were no wild
0E1C      ;          (sub)directories in the related-file directory specification.
0E1C      ;
0E1C      ;       2. Legal wildcards ("*" and "...") can only trail and are mutually
0E1C      ;          exclusive.
0E1C      ;
0E1C      ;       3. The format of partially wild output-file directory specifications
0E1C      ;          and their related-file directory specifications must be the SAME
0E1C      ;          (UFD and UFD or UIC and UIC).
0E1C      ;
0E1C      ; MOVE_RELATED substitutes subdirectories from the related file name
0E1C      ; block for subdirectories in the expanded file name string. The
0E1C      ; algorithm is as follows:
0E1C      ;
0E1C      ;       1. Get 1st related directory we're supposed to move
0E1C      ;       2. Move it
0E1C      ;       3. While there are related directories move them
0E1C      ;
0E1C      ; Calling Sequence:
0E1C      ;
0E1C      ;       BSBW    STICKY_DIR
0E1C      ;
0E1C      ; Input Parameters:
0E1C      ;       none
0E1C      ;
```

```
                     0E1C  3314 ; Implicit Input:
                     0E1C  3315 ;
                     0E1C  3316 ;        R6 - descriptor of RLF string
                     0E1C  3317 ;        R7 - descriptor of RLF string
                     0E1C  3318 ;
                     0E1C  3319 ;        R8 - fab
                     0E1C  3320 ;        R10 - fwa
                     0E1C  3321 ;        R11 - FSCB
                     0E1C  3322 ;
                     0E1C  3323 ; Output:
                     0E1C  3324 ;
                     0E1C  3325 ;        R0 - STATUS
                     0E1C  3326 ;
                     0E1C  3327 ; Implicit Output:
                     0E1C  3328 ;
                     0E1C  3329 ;        Wild directory descriptors may be changed to actual values
                     0E1C  3330 ;        FWA$V_DIR_LVLS is appropriately updated
                     0E1C  3331 ;        FWA$B_DIRLEN is appropriately updated
                     0E1C  3332 ;
                     0E1C  3333 ;        Directory descriptor is replaced with either a related subdirectory
                     0E1C  3334 ;        descriptor or a series of subdirectory descriptors
                     0E1C  3335 ;
                     0E1C  3336 ;        FWA$V_DIR_LVLS is updated to contain the number of directories
                     0E1C  3337 ;        now in the output-file.
                     0E1C  3338 ;
                     0E1C  3339 ;--
                     0E1C  3340
                     0E1C  3341         ASSUME  FWA$V_WILD_UFD  EQ      <FWA$V_DIR1+8>
                     0E1C  3342         ASSUME  FWA$B_DIRWCFLGS EQ      <FWA$B_DIRFLGS+1>
                     0E1C  3343
                     0E1C  3344 STICKY_DIR:
          59     DD  0E1C  3345         PUSHL   R9                      ; save ifb ptr
     6A   1C     E1  0E1E  3346         BBC     #FWA$V_WILD_DIR,(R10),- ; If there are no wild directories
          41         0E21  3347                 38$                     ; there is nothing to do so go
                     0E22  3348                                         ; exit with success
                     0E22  3349
                     0E22  3350 ;
                     0E22  3351 ;   initialize some variables and obtain the RLF NAM FNB bits
                     0E22  3352 ;
                     0E22  3353
          59     D4  0E22  3354 10$:    CLRL    R9                      ; R9 <- index of first wild dir in FWA
54   0130 8F     3C  0E24  3355         MOVZWL  #FWA$Q_DIR1,R4          ; R4 <- disp to directory descriptors
51   00A8 CA     D0  0E29  3356         MOVL    FWA$L_SLBH_PTR(R10),R1  ; point to current SLBH
                     0E2E  3357
                     0E2E  3358 ;
                     0E2E  3359 ; implement the syntax rules restricting partially wild UFD output-file
                     0E2E  3360 ; directory specifications to UFD formatted related-file directory
                     0E2E  3361 ; specifications and partially/totally wild UIC output-file directory
                     0E2E  3362 ; specifications to UIC formatted related-file directory specifications.
                     0E2E  3363 ; also, if the output-file directory is totally wild ([*] or [*...]) and the
                     0E2E  3364 ; related-file directory specification is in UIC format, mark the output-file
                     0E2E  3365 ; directory specification as being in UIC format.
                     0E2E  3366 ;
                     0E2E  3367
  08 6A   1B     E1  0E2E  3368         BBC     #FWA$V_GRPMBR,(R10),20$ ; if the output-file is in UIC format
          13     E1  0E32  3369         BBC     #NAM$V_GRP_MBR,-        ; but the related-file is not, then
  29 10 A1           0E34  3370                 SLBH$L_NAM_FNB(R1),35$  ; go exit with an error
```

```
            008F   31  0E37  3371         BRW     80$                     ; both are in UIC format
                       0E3A  3372
              13   E1  0E3A  3373 20$:    BBC     #NAM$V_GRP_MBR,-        ; if the related-file is in UIC format
         08 10 A1      0E3C  3374                 SLBH$L_NAM_FNB(R1),30$  ;  but the output-file is not, the
      1D 6A   28   E1  0E3F  3375         BBC     #FWA$V_WILD_UFD,(R10),35$; latter must be completely wild
                       0E43  3376         SSB     #FWA$V_GRPMBR,(R10)     ; indicate output-file in UIC format
                       0E47  3377
                       0E47  3378 ;
                       0E47  3379 ; if the first output-file directory is wild and is the correct type (ie - ''*'')
                       0E47  3380 ; but subdirectories follow, return an error
                       0E47  3381 ;
                       0E47  3382
      1B 6A   28   E1  0E47  3383 30$:    BBC     #FWA$V_WILD_UFD,(R10),40$; if the first directory is wild,
            0124   30  0E4B  3384         BSBW    CHECK_WLD_FIELD         ;  it must be of the correct type
              10   12  0E4E  3385         BNEQ    35$                     ;  and must not be followed by any
      0C 6A   21   E0  0E50  3386         BBS     #FWA$V_DIR2,(R10),35$   ;  directory or it is an error
                       0E54  3387
                       0E54  3388 ;
                       0E54  3389 ; if the output-file directory is totally wild and of the correct syntax
                       0E54  3390 ; (ie [*] or [*...]), setup to substitute and go substitute in the entire
                       0E54  3391 ; directory specification of the related-file. if however, the related-file
                       0E54  3392 ; directory specification is [] then there are no related (sub)directories
                       0E54  3393 ; to transfer, so immediately return success.
                       0E54  3394 ;
                       0E54  3395
              14   E0  0E54  3396         BBS     #FSCB$V_NULL,-          ; if the related-file is null
         0A 24 AB      0E56  3397                 FSCB$Q_DIRECTORY(R11),38$
                       0E59  3398
           2E AA   94  0E59  3399         CLRB    FWA$B_DIRLEN(R10)       ; zero the directory length
              5C   D4  0E5C  3400         CLRL    AP                      ; start with the first related subdir
              27   11  0E5E  3401         BRB     55$                     ; go do it
                       0E60  3402
            009D   31  0E60  3403 35$:    BRW     S_ERR
                       0E63  3404
            00A1   31  0E63  3405 38$:    BRW     S_SUC
```

```
                          0E66  3407 ;
                          0E66  3408 ; The first directory in the output-file is not wild
                          0E66  3409 ;
                          0E66  3410 ; Find the first wild (sub)directory in the related-file then prepare to
                          0E66  3411 ; find the number of trailing asterisks or the position of the trailing
                          0E66  3412 ; ellipsis in the output-file directory specification.
                          0E66  3413 ;
                          0E66  3414
              18     EA   0E66  3415 40$:     FFS     #NAM$V_WILD_UFD,-            ; find the offset in the filename bit
              08          0E68  3416                  #FWA$C_MAXSUBDIR+1,-         ;   field to the first related-file wild
      50   10 A1          0E69  3417                  SLBH$L_NAM_FNB(R1),R0        ;   directory
   5C    50   18     C3   0E6C  3418          SUBL3   #NAM$V_WILD_UFD,R0,AP        ; compute index of the RLF wild dir
                          0E70  3419                                               ; NOTE: AP = MAXSUBDIR+1 if none found
              51     D4   0E70  3420          CLRL    R1                           ; init wild output directory counter
                          0E72  3421
                          0E72  3422 ;
                          0E72  3423 ; When an ellipsis is encountered in the scan of the output-file directory
                          0E72  3424 ; specification, if legal wildcards had already been encountered ( R1 <> 0)
                          0E72  3425 ; or if any subdirectory follows, an error is returned.
                          0E72  3426 ; Otherwise, move the related-file directory subtree starting with the first
                          0E72  3427 ; wild (sub)directory in the related-file directory specification to the
                          0E72  3428 ; output-file directory specification starting at the position of the trailing
                          0E72  3429 ; ellipsis.
                          0E72  3430 ;
                          0E72  3431
   16 6A44    10     E5   0E72  3432 50$:     BBCC    #FSCB$V_ELIPS,(R10)[R4],-
                          0E77  3433                  60$                          ; look at next dir if no ellipsis
                          0E77  3434                                               ;  and clear the flags if there is
              51     D5   0E77  3435          TSTL    R1                           ; error if any wild directories
              E5     12   0E79  3436          BNEQ    35$                          ; had already been encountered
              59     D6   0E7B  3437          INCL    R9                           ; correct R9 for ellipsis position
   DE 04 AA   59     E0   0E7D  3438          BBS     R9,FWA$B_DIRFLGS(R10),35$          ; error if any directory follows
                          0E82  3439                                               ;  the ellipsis in the output-file
                          0E82  3440
         5C   08     91   0E82  3441          CMPB    #FWA$C_MAXSUBDIR+1,AP        ; if there are no wild related-file
              34     13   0E85  3442          BEQL    75$                          ;  (sub)directories, then return success
                          0E87  3443
         54   08     9A   0E87  3444 55$:     MOVZBL  #FWA$C_MAXSUBDIR+1,R4        ; transfer as many dirs as required
            0081     31   0E8A  3445          BRW     MOVE_RELATED                 ; go do it
```

```
                            0E8D  3447
                            0E8D  3448 ;
                            0E8D  3449 ; if we have completely scanned the output-file directory specification, we
                            0E8D  3450 ; go determine how many trailing asterisk were encountered, if any. otherwise,
                            0E8D  3451 ; we test the next subdirectory to see if its wild. if it is, we go check its
                            0E8D  3452 ; legality. if it is not, but a wild subdirectory preceeded it, return an
                            0E8D  3453 ; error. finally, if the next directory is not wild and no wild subdirectory
                            0E8D  3454 ; preceeded it, go check whether an ellipsis follows.
                            0E8D  3455 ;
                            0E8D  3456
                   59  D6   0E8D  3457 60$:      INCL      R9                            ; Offset for next wild dir bit
             54    08  C0   0E8F  3458           ADDL      #8,R4                         ; Disp for next descriptor
                            0E92  3459
             08    59  91   0E92  3460           CMPB      R9,#FWA$C_MAXSUBDIR+1         ; if looked at all directories
                   1D  13   0E95  3461           BEQL      70$                           ; then go decide what must be done
                 6A44  95   0E97  3462           TSTB      (R10)[R4]                     ; Is descriptor null ?
                   18  13   0E9A  3463           BEQL      70$                           ; Yes - done with out dir
                            0E9C  3464
       07 05 AA    59  E0   0E9C  3465           BBS       R9,FWA$B_DIRWCFLGS(R10),65$ ; if subdir is wild, check its legality
                   51  D5   0EA1  3466           TSTL      R1                            ; if subdirectory is nonwild but the
                   CD  13   0EA3  3467           BEQL      50$                           ;  preceeding directory was, go return
                 0058  31   0EA5  3468 63$:      BRW       S_ERR                         ; an error, else go test for an ...
                            0EA8  3469
                            0EA8  3470 ;
                            0EA8  3471 ; when a wild subdirectory is encountered its legality is checked. if it is
                            0EA8  3472 ; of the correct type (ie "*") we continue the scan of the output-file
                            0EA8  3473 ; directory specification by checking whether an ellipsis follows. otherwise,
                            0EA8  3474 ; an error is returned.
                            0EA8  3475 ;
                            0EA8  3476
                 00C7  30   0EA8  3477 65$:      BSBW      CHECK_WLD_FIELD               ; Is wild dir the legal type ?
                   F8  12   0EAB  3478           BNEQ      63$                           ; No  - exit with error
                   51  D6   0EAD  3479           INCL      R1                            ; yes - incr wild dir count
                2E AA  97   0EAF  3480           DECB      FWA$B_DIRLEN(R10)             ; Decrement the total directory length
                   BE  11   0EB2  3481           BRB       50$                           ; continue looking at directories
```

```
                        0EB4   3483 ;
                        0EB4   3484 ; the entire output-file directory specification has been scanned with
                        0EB4   3485 ; no trailing ellipis found
                        0EB4   3486 ;
                        0EB4   3487 ; if no trailing asterisks were encountered, go return success,
                        0EB4   3488 ; otherwise, compute the wild directory bit and directory descriptor
                        0EB4   3489 ; displacements for the first asterisk encountered, and go move the
                        0EB4   3490 ; required number of related-file directories.
                        0EB4   3491 ;
                        0EB4   3492
              51   D5   0EB4   3493 70$:    TSTL    R1                              ; if no trailing wild dirs found
              4F   13   0EB6   3494         BEQL    S_SUC                           ; then return success
                        0EB8   3495
                        0EB8   3496 ;
                        0EB8   3497 ; if there arn't any wild related-file (sub)directories, then return
                        0EB8   3498 ;
                        0EB8   3499
         5C   08   91   0EB8   3500         CMPB    #FWA$C_MAXSUBDIR+1,AP           ; if there are no wild related-file
              4A   13   0EBB   3501 75$:    BEQL    S_SUC                           ; then return success
                        0EBD   3502
              1D   EF   0EBD   3503         EXTZV   #FWA$V_DIR_LVLS,-               ; extract the number of subdirectory
    59   6A   03        0EBF   3504                 #FWA$S_DIR_LVLS,(R10),R9;        levels in the output-file
         59   51   C2   0EC2   3505         SUBL2   R1,R9                           ; subtract the trailing asterisks
              59   D6   0EC5   3506         INCL    R9                              ; need to add one to get to the subdir
              32   11   0EC7   3507         BRB     110$                            ; go perform the substitution
```

```
                    0EC9
                    0EC9     ;
                    0EC9     ; for UIC formatted output-file directories, preset the number of the
                    0EC9     ; related-file directory to be first to be moved to 0 (the group), the number
                    0EC9     ; of directories to be moved to 1 (just the group part), and predecrement the
                    0EC9     ; total directory count appropriately.
                    0EC9     ;
                    0EC9
         5C    D4   0EC9     80$:    CLRL    AP                      ; preset wild related-file dir # to 0
      51 01    D0   0ECB             MOVL    #1,R1                   ; preset number of dirs to move to 1
      2E AA    97   0ECE             DECB    FWA$B_DIRLEN(R10)       ; decrement total directory length
                    0ED1
                    0ED1     ;
                    0ED1     ; if the group part of the UIC formatted output-file directory is wild, make
                    0ED1     ; sure its of a valid type, and then set the number of subdirectory levels kept
                    0ED1     ; within the FWA to 0 so that the first directory to be replaced will be the
                    0ED1     ; group part of the output-file.
                    0ED1     ;
                    0ED1
         28    E1   0ED1             BBC     #FWA$V_WILD_GRP,-       ; if group of UIC format output-file
      1B 6A         0ED3                     (R10),90$               ; dir is not wild, go check member part
       009A    30   0ED5             BSBW    CHECK_WLD_FIELD         ; if wild group part of dir is not
         26    12   0ED8             BNEQ    S_ERR                   ; valid then return an error
                    0EDA
                    0EDA     ;
                    0EDA     ; if both the group and member parts of the UIC formatted output-file directory
                    0EDA     ; specification are wild (ie [*,*]), then make sure both are of the correct
                    0EDA     ; type, increment the number of directories to be moved and decrement the
                    0EDA     ; directory length appropriately, and then go substitute in both and member
                    0EDA     ; parts from the UIC formatted related-file. if just the group part is wild, go
                    0EDA     ; substitute just the group part from the related-file.
                    0EDA     ;
                    0EDA
         29    E1   0EDA             BBC     #FWA$V_WILD_MBR,-       ; if output-file dir is not [*,*]
      1D 6A         0EDC                     (R10),110$              ; then go setup transfer as is
   54    08    C0   0EDE             ADDL2   #8,R4                   ; increment dir descriptor displacement
       008E    30   0EE1             BSBW    CHECK_WLD_FIELD         ; if wild member part of dir is not
         1A    12   0EE4             BNEQ    S_ERR                   ;  restoring saved displacement values
   54    08    C2   0EE6             SUBL2   #8,R4                   ; restore dir descriptor displacement
         51    D6   0EE9             INCL    R1                      ; increment count of wild directories
      2E AA    97   0EEB             DECB    FWA$B_DIRLEN(R10)       ; decrease total directory length
         0B    11   0EEE             BRB     110$                    ; go transfer both group and member
```

```
                    0EF0  3552
                    0EF0  3553 ;
                    0EF0  3554 ; if just the member part of the UIC formatted output-file directory
                    0EF0  3555 ; specification is wild, make sure its of the correct type, and that the member
                    0EF0  3556 ; part of the output-file directory is to be replaced by the member part of the
                    0EF0  3557 ; the related-file directory specification before going and performing the
                    0EF0  3558 ; substitution.
                    0EF0  3559 ;
                    0EF0  3560
          5C    D6  0EF0  3561 90$:     INCL    AP                              ; related-file member to be transfered
          59    D6  0EF2  3562          INCL    R9                              ; incrm wild bit disp to wild member
     54   08    C0  0EF4  3563          ADDL2   #8,R4                           ; incrm descriptor to UIC member slot
          79    10  0EF7  3564          BSBB    CHECK_WLD_FIELD                 ; if wild group part of dir is not
          05    12  0EF9  3565          BNEQ    S_ERR                           ; valid then return an error
                    0EFB  3566
                    0EFB  3567 ;
                    0EFB  3568 ; when either UIC formatted output-file directory specifications or partially
                    0EFB  3569 ; wild directory specifications in UFD formatted are involved, setup to move
                    0EFB  3570 ; the required number of related-file directories (or directory parts in the
                    0EFB  3571 ; case of a UIC formatted output-file directory), and go move them. return
                    0EFB  3572 ; success when the substitution is completed.
                    0EFB  3573 ;
                    0EFB  3574
     54   51    D0  0EFB  3575 110$:    MOVL    R1,R4                           ; number of directories to move
          0E    11  0EFE  3576 120$:    BRB     MOVE_RELATED                    ; go do it
                    0F00  3577
                    0F00  3578 ;
                    0F00  3579 ; exit with the appropriate status
                    0F00  3580 ;
                    0F00  3581
                    0F00  3582 S_ERR:   RMSERR  DIR                             ; error
          03    11  0F05  3583          BRB     S_EXIT
                    0F07  3584
                    0F07  3585 S_SUC:   RMSSUC                                  ; success
          59  8ED0  0F0A  3586 S_EXIT:  POPL    R9                              ; restore ifab
                05  0F0D  3587          RSB                                     ; exit
```

```
                                 0F0E  3589 ; Working Registers for MOVE_RELATED:
                                 0F0E  3590 ;
                                 0F0E  3591 ;           R4      - number of related subdirectories to move
                                 0F0E  3592 ;           R9      - index of the output subdirectory to start with
                                 0F0E  3593 ;           AP      - index of the related subdirectory to start with
                                 0F0E  3594 ;
                                 0F0E  3595
                                 0F0E  3596 MOVE_RELATED:
          03 AB     5C     91    0F0E  3597         CMPB    AP,FSCB$B_DIRS(R11)
                    26     1E    0F12  3598         BGEQU   40$
                                 0F14  3599
                                 0F14  3600 ;
                                 0F14  3601 ; Get the related directory we're supposed to move
                                 0F14  3602 ;
                                 0F14  3603
             07     59     91    0F14  3604 10$:    CMPB    R9,#FWA$C_MAXSUBDIR               ; too many subdirs ?
                    E7     1A    0F17  3605         BGTRU   S_ERR                             ; yes - error
     56    00C4 CB4C      7D    0F19  3606         MOVQ    FSCB$Q_DIRECTORY1(R11)[AP],R6    ; get the first rlf directory
          2E AA     56     80    0F1F  3607         ADDB2   R6,FWA$B_DIRLEN(R10)             ; accumulate the total length
                    DB     1F    0F23  3608         BCS     S_ERR                             ; if overflow the byte
                                 0F25  3609
                                 0F25  3610 ;
                                 0F25  3611 ; We have to make this a subroutine call since CHECK_FIELD returns
                                 0F25  3612 ; one level up if there is an error, which is bad news since we
                                 0F25  3613 ; have pushed registers on the stack in this routine
                                 0F25  3614 ;
                                 0F25  3615
                    32     10    0F25  3616         BSBB    MOVE_REL_FIELD                  ; copy this field using CHECK_FIELD
                 D6 50     E9    0F27  3617         BLBC    R0,S_ERR                        ; quit on error
                                 0F2A  3618
                    54     D7    0F2A  3619         DECL    R4                              ; decrm # of dirs to move
                    17     13    0F2C  3620         BEQL    50$                             ; exit if no more to be moved
                                 0F2E  3621
                    5C     D6    0F2E  3622         INCL    AP                              ; get the next rlf directory index
          03 AB     5C     91    0F30  3623         CMPB    AP,FSCB$B_DIRS(R11)             ; have we moved the last one ?
                    04     13    0F34  3624         BEQL    40$                             ; yes, exit
                    59     D6    0F36  3625         INCL    R9                              ; get the next output directory
                    DA     11    0F38  3626         BRB     10$                             ;  and continue
                                 0F3A  3627
                                 0F3A  3628 ;
                                 0F3A  3629 ; There may be more related-file subdirectories to move because R9 > 0
                                 0F3A  3630 ; but no related-file subdirectories to move.
                                 0F3A  3631 ;
                                 0F3A  3632 ; This situation can arise because STICKY_DIR requests 8 directories to be
                                 0F3A  3633 ; moved when it encounters a trailing ellipsis, or the output-file directory
                                 0F3A  3634 ; specification consists of [*] or [*...], and because there may be more
                                 0F3A  3635 ; trailing asterisks in the output-file directory specification then there
                                 0F3A  3636 ; are related-file directories to substitute. When the latter case is
                                 0F3A  3637 ; encountered, adjust the number of subdirectories so that FWA$V_DIR_LVLS
                                 0F3A  3638 ; is set correctly remaining asterisks.
                                 0F3A  3639 ;
                                 0F3A  3640
     51     59     21     C1    0F3A  3641 40$:    ADDL3   #FWA$V_DIR2,R9,R1               ; check to see whether there are any
          03 6A     51     E1    0F3E  3642         BBC     R1,(R10),50$                    ;  trailing output-file directories
             59     54     C0    0F42  3643         ADDL2   R4,R9                           ; add what is left from what was
                                 0F45  3644                                                 ;  required to be copied
                                 0F45  3645
```

```
                           0F45  3646 ;
                           0F45  3647 ; exit with success
                           0F45  3648 ;
                           0F45  3649
        09 6A    1B    E1  0F45  3650 50$:      BBC     #FWA$V_GRPMBR,(R10),60$ ; how do we set lvls?
                           0F49  3651           SSB     #FSCB$V_GRPMBR,FWA$Q_DIR1(R10)  ; reset UIC format flag
           59    01    D0  0F4F  3652           MOVL    #1,R9                         ; UIC treated as having 2 levels
                           0F52  3653
           1D    59    F0  0F52  3654 60$:      INSV    R9,#FWA$V_DIR_LVLS,-          ; normal number of subdirs
           6A    03        0F55  3655                   #FWA$S_DIR_LVLS,(R10)
                 AE    11  0F57  3656           BRB     S_SUC
                           0F59  3657
                           0F59  3658 ;
                           0F59  3659 ; We have to make this a subroutine call since CHECK_FIELD returns
                           0F59  3660 ; one level up if there is an error, which is bad news since we
                           0F59  3661 ; have pushed registers on the stack in this routine
                           0F59  3662 ;
                           0F59  3663
                           0F59  3664 MOVE_REL_FIELD:
                           0F59  3665           SSB     R9,FWA$B_DIRWCFLGS(R10) ; set wild dir bit
     50    59    20    C1  0F5E  3666           ADDL3   #FWA$V_DIR1,R9,R0       ; subdirectory field
           51    27    9A  0F62  3667           MOVZBL  #FWA$C_MAXNAME,R1       ; maximum directory size
     55  0130  CA49    7E  0F65  3668           MOVAQ   FWA$Q_DIR1(R10)[R9],R5  ; directory descriptor
               FE17    30  0F6B  3669           BSBW    CHECK_FIELD             ; move this directory
           50    01    9A  0F6E  3670           MOVZBL  #1,R0                   ; returning here is ok
                       05  0F71  3671           RSB
                           0F72  3672
```

```
                        0F72  ;
                        0F72  ; CHECK_WLD_FIELD
                        0F72  ;
                        0F72  ; This subroutine checks a wild field in the FWA (directory, subdirectory,
                        0F72  ; subdirectory part in the case of UIC formatted directories, filename, or
                        0F72  ; type) to make sure that it is in legal format. Legal format is "*" and
                        0F72  ; only "*".
                        0F72  ;
                        0F72  ; Input Parameters:
                        0F72  ;
                        0F72  ;         R4      - offset to field descriptor, the contents of which
                        0F72  ;                   are to be checked for wild validity
                        0F72  ;         R10     - fwa
                        0F72  ;
                        0F72  ; Output:
                        0F72  ;
                        0F72  ;         V bit   - SET   if descriptor contains a legally formatted wild field
                        0F72  ;                   CLEAR otherwise
                        0F72  ;
                        0F72  ;         R0      - destroyed
                        0F72  ;
                        0F72
                        0F72 CHECK_WLD_FIELD:
50     5A     54     C1 0F72          ADDL3   R4,R10,R0                       ; address of descriptor
       60     01     B1 0F76          CMPW    #1,(R0)                         ; check size
              04     12 0F79          BNEQ    10$                             ; bad
   04 B0      2A     91 0F7B          CMPB    #^A/*/,@4(R0)                   ; right character
                     05 0F7F 10$:     RSB                                     ;
                        0F80
                        0F80          .END                                    ; End of module
```

```
$$.PSECT_EP                 = 00000000                FNMERR                        000008A4 R      01
$$ARGS                      = 00000006                FOP                         = 00000020
$$RMSTEST                   = 0000001A                FSCB$B_DIRS                 = 00000003
$$RMS_PBUGCHK               = 00000010                FSCB$B_FLDFLAGS             = 00000000
$$RMS_TBUGCHK               = 00000008                FSCB$B_NODES                = 00000001
$$RMS_UMODE                 = 00000004                FSCB$B_ROOTS                = 00000002
$$T1                        = 0000001C                FSCB$C_BLN                  = 00000104
ALLOCATE_SLB                  00000CA1 R      01      FSCB$C_MAXDIR               = 00000008
APPEND_NODESPEC               0000082C R      01      FSCB$K_BLN                  = 00000104
CHECK_DEVICE                  0000056D R      01      FSCB$M_DEVICE               = 00000002
CHECK_FIELD                   00000D85 R      01      FSCB$M_DIRECTORY            = 00000008
CHECK_NAME                    00000867 R      01      FSCB$M_ELIPS                = 00010000
CHECK_NODE                    00000580 R      01      FSCB$M_MINUS                = 00800000
CHECK_ROOT                    00000C4A R      01      FSCB$M_NAME                 = 00000010
CHECK_TYPE                    0000048C R      01      FSCB$M_NODE                 = 00000001
CHECK_VERSION                 000004AA R      01      FSCB$M_ROOT                 = 00000004
CHECK_WLD_FIELD               00000F72 R      01      FSCB$M_WILD                 = 00020000
CHKDIR                        000003B0 R      01      FSCB$Q_DEVICE               = 00000014
CHKDUP                        00000E0A R      01      FSCB$Q_DIRECTORY            = 00000024
CHKERR                        00000E16 R      01      FSCB$Q_DIRECTORY1           = 000000C4
CHKNAMBLK                     0000021F R      01      FSCB$Q_DIRECTORY2           = 000000CC
CHKRTN                        00000E07 R      01      FSCB$Q_FILESPEC             = 00000004
CHKSUC                        00000E00 R      01      FSCB$Q_NAME                 = 0000002C
COPY_DIR                      00000999 R      01      FSCB$Q_NODE                 = 0000000C
CPYDSC                        00000E04 R      01      FSCB$Q_NODE1                = 00000044
DEFAULT_DIR                   00000A0D R      01      FSCB$Q_NODE2                = 0000004C
DIRXIT                        000009F7 R      01      FSCB$Q_ROOT1                = 00000084
DIRXIT1                       0000092E R      01      FSCB$Q_ROOT8                = 000000BC
DONTUSE                       00000BE6 R      01      FSCB$Q_TYPE                 = 00000034
ENDPRS                        000003C3 R      01      FSCB$Q_VERSION              = 0000003C
ERRDIR                        0000098B R      01      FSCB$V_ACS                  = 00000012
ERRDIR1                       000008DD R      01      FSCB$V_CONCEAL              = 00000018
ERRDNA                        0000023E R      01      FSCB$V_DEVICE               = 00000001
ERRDVI                        0000022C R      01      FSCB$V_DIRECTORY            = 00000003
ERRFNA                        00000238 R      01      FSCB$V_ELIPS                = 00000010
ERRLNE                        00000B39 R      01      FSCB$V_GRPMBR               = 00000016
ERRNOD                        000006A5 R      01      FSCB$V_MFD                  = 00000019
ERRNOD1                       000006A2 R      01      FSCB$V_MINUS                = 00000017
ERRNOD2                       00000801 R      01      FSCB$V_NAME                 = 00000004
ERRNOD3                       000007FE R      01      FSCB$V_NODE                 = 00000000
ERRRST                        00000232 R      01      FSCB$V_NULL                 = 00000014
ESCAPE                      = 0000001B                FSCB$V_PWD                  = 00000015
ESCAPE_CHAR                   00000CF5 R      01      FSCB$V_QUOTED               = 00000013
EXE$UPCASE_DAT                ********    X   01      FSCB$V_ROOT                 = 00000002
EXPAND_NAME                   0000017C R      01      FSCB$V_TYPE                 = 00000005
EXPARGL                       00000146 R      01      FSCB$V_VERSION              = 00000006
EXTRAN                        00000BF0 R      01      FSCB$V_WILD                 = 00000011
FAB$B_DNS                   = 00000035                FWA$B_BUFFLG                = 0000009C
FAB$B_FNS                   = 00000034                FWA$B_DIRFLGS               = 00000004
FAB$L_DNA                   = 00000030                FWA$B_DIRLEN                = 0000002E
FAB$L_FNA                   = 0000002C                FWA$B_DIRTERM               = 0000000A
FAB$L_FOP                   = 00000004                FWA$B_DIRWCFLGS             = 00000005
FAB$L_NAM                   = 00000028                FWA$B_ESCFLG                = 0000000C
FAB$V_NAM                   = 00000018                FWA$B_FLDFLGS               = 00000001
FAB$V_OFP                   = 0000001D                FWA$B_LEVEL                 = 000000C3
FILE_OR_LOGNAME               00000541 R      01      FWA$B_LNFLGS                = 00000006
FINISH_PARSE                  000004C8 R      01      FWA$B_PASSFLGS              = 00000000
```

```
FWA$B_ROOTERM                    = 0000000B          FWA$T_SLB                        = 000000B8
FWA$B_SUBNODCNT                  = 0000002F          FWA$V_CONCEAL_DEV                = 00000039
FWA$B_UNDER_NOD                  = 000007E8          FWA$V_DEVICE                     = 0000000F
FWA$B_WILDFLGS                   = 00000002          FWA$V_DFLT_MFD                   = 0000003B
FWA$B_XLTMODE                    = 0000009D          FWA$V_DIR                        = 0000000E
FWA$C_ALL                        = 000000F8          FWA$V_DIR1                       = 00000020
FWA$C_ALLPASS                    = 00000019          FWA$V_DIR2                       = 00000021
FWA$C_MAXDIRLEN                  = 000000FF          FWA$V_DIR_LVLS                   = 0000001D
FWA$C_MAXLNDNAM                  = 0000000F          FWA$V_DUPOK                      = 00000000
FWA$C_MAXNAME                    = 00000027          FWA$V_EXP_DEV                    = 00000017
FWA$C_MAXNODLST                  = 0000007F          FWA$V_EXP_DIR                    = 00000016
FWA$C_MAXNODNAM                  = 00000006          FWA$V_EXP_NAME                   = 00000012
FWA$C_MAXQUOTED                  = 000000FF          FWA$V_EXP_NODE                   = 00000006
FWA$C_MAXSUBDIR                  = 00000007          FWA$V_EXP_ROOT                   = 0000003C
FWA$C_MAXSUBNOD                  = 00000007          FWA$V_EXP_TYPE                   = 00000011
FWA$C_MAXTYPE                    = 00000027          FWA$V_EXP_VER                    = 00000010
FWA$C_MAXVER                     = 00000006          FWA$V_FNA_PASS                   = 00000004
FWA$L_BUF_PTR                    = 00000050          FWA$V_GRPMBR                     = 0000001B
FWA$L_ESCSTRING                  = 0000000C          FWA$V_LOGNAME                    = 00000030
FWA$L_IMPURE_AREA                = 00000054          FWA$V_NAME                       = 0000000D
FWA$L_SLBH_BLINK                 = 000000B4          FWA$V_NAM_DVI                    = 00000005
FWA$L_SLBH_FLINK                 = 000000B0          FWA$V_NOCOPY                     = 00000001
FWA$L_SLBH_PTR                   = 000000A8          FWA$V_NODE                       = 00000019
FWA$L_SLB_PTR                    = 000000AC          FWA$V_QUOTED                     = 0000001A
FWA$L_XLTBUFF1                   = 000000A0          FWA$V_RLF_PASS                   = 00000003
FWA$L_XLTBUFF2                   = 000000A4          FWA$V_ROOT_DIR                   = 0000003A
FWA$M_DEVICE                     = 00008000          FWA$V_SLPRESENT                  = 00000038
FWA$M_DIR                        = 00004000          FWA$V_SL_PASS                    = 00000002
FWA$M_DUPOK                      = 00000001          FWA$V_SYNTAX_CHK                 = 00000036
FWA$M_EXP_DEV                    = 00800000          FWA$V_TYPE                       = 0000000C
FWA$M_EXP_DIR                    = 00400000          FWA$V_VERSION                    = 0000000B
FWA$M_EXP_NODE                   = 00000040          FWA$V_WC_NAME                    = 00000015
FWA$M_EXP_TYPE                   = 00020000          FWA$V_WC_TYPE                    = 00000014
FWA$M_EXP_VER                    = 00010000          FWA$V_WC_VER                     = 00000013
FWA$M_FNA_PASS                   = 00000010          FWA$V_WILDCARD                   = 00000018
FWA$M_NAM_DVI                    = 00000020          FWA$V_WILD_DIR                   = 0000001C
FWA$M_RLF_PASS                   = 00000008          FWA$V_WILD_GRP                   = 00000028
FWA$Q_CDIR1                      = 000000F0          FWA$V_WILD_MBR                   = 00000029
FWA$Q_CDIR8                      = 00000128          FWA$V_WILD_UFD                   = 00000028
FWA$Q_CONCEAL_DEV                = 000000E8          FWA$W_XLTSIZ                     = 0000009E
FWA$Q_DEVICE                     = 000000E0          GETSPC                             00000244 R       01
FWA$Q_DIR1                       = 00000130          GET_NEXT                           000009BF R       01
FWA$Q_DIR2                       = 00000138          GRPMBR                             000008E0 R       01
FWA$Q_LOGNAM                     = 000000D0          HAVEONE                            00000B7F R       01
FWA$Q_NAME                       = 00000170          HOR_TAB                          = 00000009
FWA$Q_NODE                       = 000000D8          IFB$B_MODE                       = 0000000A
FWA$Q_NODE1                      = 000001B4          IFB$L_NWA_PTR                    = 0000003C
FWA$Q_TYPE                       = 00000178          IFB$V_CREATE                     = 00000032
FWA$Q_VERSION                    = 00000180          INSERT_MFD                         000009FB R       01
FWA$S_DEVICEBUF                  = 000000FF          INTERMED_NODE                      000006AE R       01
FWA$S_DIR_LVLS                   = 00000003          LNM$C_MAXDEPTH                   = 0000000A
FWA$T_ITMLST                     = 0000005C          LNM$C_NAMLENGTH                  = 000000FF
FWA$T_ITM_ATTR                   = 00000068          LNM$M_CASE_BLIND                 = 02000000
FWA$T_ITM_INDEX                  = 0000005C          LNM$V_CONCEALED                  = 00000008
FWA$T_ITM_MAX_INDEX              = 00000080          LNM$V_EXISTS                     = 0000000A
FWA$T_ITM_STRING                 = 00000074          LNM$V_TERMINAL                   = 00000009
FWA$T_NODEBUF                    = 000007E9          LNM_ATTR                           00000A93 R       01
```

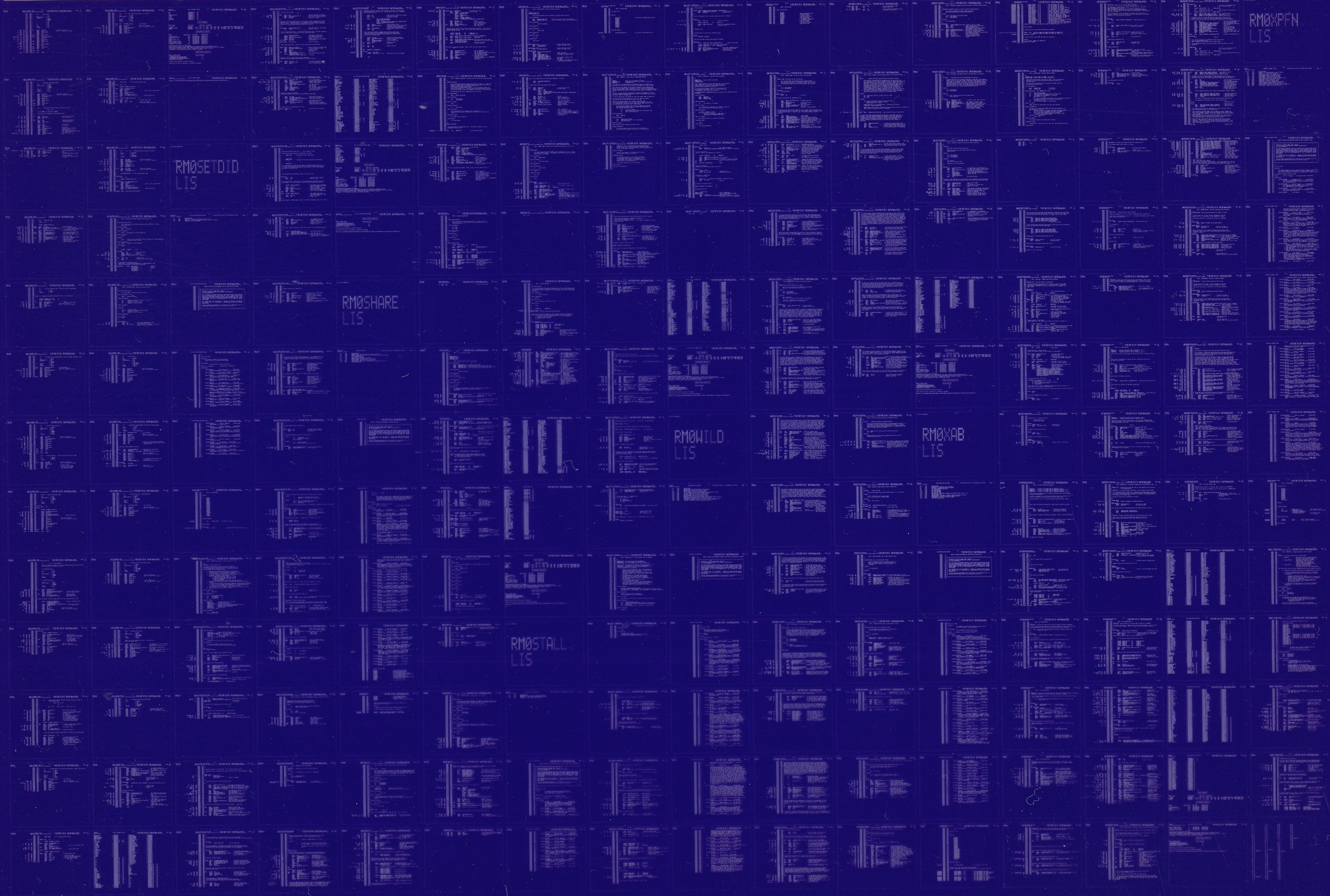
```
LOOP                0000094F R       01    NWA$L_XLTMAXINDX    00000234
MFD_ASCII           00000991 R       01    NWA$L_XLTSIZ        00000230
MOVE_RELATED        00000F0E R       01    NWA$Q_ACS           00000244
MOVE_REL_FIELD      00000F59 R       01    NWA$Q_BIGBUF        00000170
MOVNXT              000000D5 R       01    NWA$Q_BLD           000000F0
NAM$B_ESL         = 0000000B               NWA$Q_FLG           00000000
NAM$B_NOP         = 00000008               NWA$Q_INODE         0000025C
NAM$B_RSL         = 00000003               NWA$Q_IOSB          000000D8
NAM$C_MAXRSS      = 000000FF               NWA$Q_LNODE         00000160
NAM$L_ESA         = 0000000C               NWA$Q_LOGNAME       0000023C
NAM$L_FNB         = 00000034               NWA$Q_NCB           00000264
NAM$L_RLF         = 00000010               NWA$Q_RCV           000000E0
NAM$L_RSA         = 00000004               NWA$Q_SAVE_DESC     00000120
NAM$T_DVI         = 00000014               NWA$Q_XLTBUF1       0000024C
NAM$V_CNCL_DEV    = 0000000C               NWA$Q_XLTBUF2       00000254
NAM$V_GRP_MBR     = 00000013               NWA$Q_XMT           000000E8
NAM$V_NODE        = 00000011               NWA$T_ACSBUF        0000026C
NAM$V_PPF         = 00000010               NWA$T_AUXBUF        000005E0
NAM$V_ROOT_DIR    = 0000000D               NWA$T_DAP           00000000
NAM$V_SEARCH_LIST = 0000000B               NWA$T_INODEBUF      000004AC
NAM$V_SYNCHK      = 00000003               NWA$T_ITM_ATTR      00000200
NAM$V_WILD_UFD    = 00000018               NWA$T_ITM_END       00000224
NAM$W_DID         = 0000002A               NWA$T_ITM_LST       00000200
NAM$W_FID         = 00000024               NWA$T_ITM_MAXINDX   00000218
NAMED_DIR           00000936 R       01    NWA$T_ITM_STRING    0000020C
NOLOGNAM            00000B55 R       01    NWA$T_NCBBUF        0000052C
NORMAL              00000B41 R       01    NWA$T_NODEBUF       00000169
NOTRAN              00000BE2 R       01    NWA$T_RCVBUF        000001A0
NT$NWA_INIT         ********     X   01    NWA$T_SCAN          00000100
NWA$B_ALLXABCNT     0000011C               NWA$T_TEMP          00000120
NWA$B_DAP_RAC       000000C9               NWA$T_XLTBUF1       000002AC
NWA$B_FILESYS       000000C5               NWA$T_XLTBUF2       000003AC
NWA$B_KEYXABCNT     0000011D               NWA$T_XMTBUF        000003C0
NWA$B_NETSTRSIZ     0000016F               NWA$W_BUILD         000000D2
NWA$B_NODBUFSIZ     00000168               NWA$W_DAPBUFSIZ     000000CA
NWA$B_ORG           000000C6               NWA$W_DIR_OFF       000000CC
NWA$B_OSTYPE        000000C4               NWA$W_DISPLAY       000000D0
NWA$B_RFM           000000C7               NWA$W_FIL_OFF       000000CE
NWA$B_RMS_RAC       000000C8               NWA$W_JNLXABJOP     0000011E
NWA$C_BLN           00000800               PARSE_CONCEAL       00000BF7 R       01
NWA$C_INODESIZ    = 00000080               PARSE_DIR           000008A7 R       01
NWA$C_MAXACS      = 0000002C               PARSE_SPECS         0000031E R       01
NWA$K_BLN           00000800               PARSE_STRING        000003D3 R       01
NWA$L_ALLXABADR     00000100               PIO$A_TRACE         ********     X   01
NWA$L_DATXABADR     00000104               PIO$GT_DDSTRING     ********     X   01
NWA$L_DEV           000000C0               PPERMFILE           00000BB0 R       01
NWA$L_FHCXABADR     00000108               PRS_ERROR           0000053D R       01
NWA$L_KEYXABADR     0000010C               PRS_EXIT            000004F4 R       01
NWA$L_MSG_MASK      000000D4               PRS_EXIT1           00000537 R       01
NWA$L_PROXABADR     00000110               PRS_EXIT2           00000539 R       01
NWA$L_RDTXABADR     00000114               PRS_LOOP            00000325 R       01
NWA$L_SAVE_FLGS     00000128               PSEUDO_CHKFLD       00000804 R       01
NWA$L_SUMXABADR     00000118               RM$CHKNAM           ********     X   01
NWA$L_THREAD        000000FC               RM$EXPSTRING        ********     X   01
NWA$L_XLTATTR       00000238               RM$FWASET           ********     X   01
NWA$L_XLTBUFFLG     0000022C               RM$GETBLK1          ********     X   01
NWA$L_XLTCNT        00000228               RM$GETPAG           ********     X   01
```

```
RM$GETSPC1                  ********   X    01      SYS$TRNLNM                  ********  GX    01
RM$RETBLK1                  ********   X    01      SYS$TRNLOG                  ********   X    01
RM$RETPAG                   ********   X    01      SYSDISKLEN                = 00000009
RM$SCAN_STRING              ********   X    01      S_ERR                       00000F00 R      01
RM$XPFN                     00000000 RG     01      S_EXIT                      00000F0A R      01
RMS$_ACS                  = 000187B4                S_SUC                       00000F07 R      01
RMS$_BUG_DDI              = 0001843C                TABNAM                      00000A97 R      01
RMS$_DEV                  = 000184C4                TABNAM_SIZE               = 0000000C
RMS$_DIR                  = 000184CC                TPT$L_NTXLATLOG             ********   X    01
RMS$_DNA                  = 000184DC                TPT$L_XLATLOG               ********   X    01
RMS$_DVI                  = 000184F4                TPT$L_XPFN                  ********   X    01
RMS$_ESA                  = 000184FC                TRAN                        00000AD5 R      01
RMS$_ESS                  = 00018504                TRANSLATE                   00000AA3 R      01
RMS$_FNA                  = 00018524                TRANSLATE_FAIL              0000070A R      01
RMS$_FNM                  = 0001852C                TRANSLATE_LOOP              000005B8 R      01
RMS$_LNE                  = 000185BC                TRNLNM                      00000B0F R      01
RMS$_NOD                  = 000185F4                TRNLOG$_ACMODE            = 00000014
RMS$_QUO                  = 00018634                TRNLOG$_DSBMSK            = 00000018
RMS$_RLF                  = 00018674                TRNLOG$_LOGNAM            = 00000004
RMS$_RST                  = 0001869C                TRNLOG$_NARGS             = 00000006
RMS$_SYN                  = 000186D4                TRNLOG$_RSLBUF            = 0000000C
RMS$_TYP                  = 000186E4                TRNLOG$_RSLLEN            = 00000008
RMS$_VER                  = 000186FC                TRNLOG$_TABLE             = 00000010
SECONDARY_NODE              0000077E R      01      UPCASE_COMPRESS             00000D15 R      01
SETNAM                      000000E3 R      01      UPDATE_SLB                  0000014B R      01
SKPDIR                      00000931 R      01      WC                        = 00000010
SLB$B_BID                 = 00000008                WCNTV_MASK                = 00000038
SLB$B_BLN                 = 00000009                WLDFNM                      000003C7 R      01
SLB$B_FLAGS               = 0000000A                WLDTYP                      000003CD R      01
SLB$B_LEVEL               = 0000000B                XITNAM                      00000145 R      01
SLB$C_BID                 = 00000029                XPFN_SPACE                = 00000508
SLB$C_BLN                 = 00000018
SLB$L_ATTR                = 00000014
SLB$L_BLINK               = 00000004
SLB$L_FLINK               = 00000000
SLB$L_INDEX               = 0000000C
SLB$L_MAX_INDEX           = 00000010
SLB$M_REALSLB             = 00000001
SLB$V_REALSLB             = 00000000
SLBH$B_BID                = 00000008
SLBH$B_BLN                = 00000009
SLBH$B_PASSFLGS           = 0000000A
SLBH$B_STR_LEN            = 0000000B
SLBH$C_BID                = 0000002B
SLBH$C_BLN                = 00000014
SLBH$L_FLINK              = 00000000
SLBH$L_NAM_FNB            = 00000010
SLBH$L_SLB_QUE            = 0000000C
SLBH$T_STRING             = 00000014
SLBH_BLINK                = 00000004
SLBH_FLINK                = 00000000
SLBH_PTR                  = FFFFFFF8
SLB_PTR                   = FFFFFFFC
SS$_NOLOGNAM              = 000001BC
SS$_NORMAL                = 00000001
STICKY_DIR                  00000E1C R      01
STUFF_NAME                  00000276 R      01
```

```
                                              +----------------+
                                              ! Psect synopsis !
                                              +----------------+

PSECT name                        Allocation            PSECT No.  Attributes
----------                        ----------            ---------  ----------
.  ABS  .                         00000000  (     0.)   00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
RMS$RMSFILENAME                   00000F80  (  3968.)   01 (  1.)    PIC   USR   CON   REL   GBL NOSHR   EXE   RD  NOWRT NOVEC BYTE
$ABS$                             00000800  (  2048.)   02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE

                                           +------------------------+
                                           ! Performance indicators !
                                           +------------------------+

Phase                          Page faults    CPU Time        Elapsed Time
-----                          -----------    --------        ------------
Initialization                          35    00:00:00.12     00:00:01.20
Command processing                     132    00:00:00.84     00:00:09.23
Pass 1                                 623    00:00:26.50     00:00:56.67
Symbol table sort                        0    00:00:03.03     00:00:03.80
Pass 2                                 642    00:00:09.61     00:00:21.54
Symbol table output                      4    00:00:00.33     00:00:00.59
Psect synopsis output                    3    00:00:00.04     00:00:00.04
Cross-reference output                   0    00:00:00.00     00:00:00.00
Assembler run totals                  1442    00:00:40.47     00:01:33.12

The working set limit was 1950 pages.
149359 bytes (292 pages) of virtual memory were used to buffer the intermediate code.
There were 110 pages of symbol table space allocated to hold 1891 non-local and 212 local symbols.
3703 source lines were read in Pass 1, producing 22 object records in Pass 2.
33 pages of virtual memory were used to define 32 macros.

                                          +--------------------------+
                                          ! Macro library statistics !
                                          +--------------------------+

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                  17
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                   1
_$255$DUA28:[SYSLIB]STARLET.MLB;2               10
TOTALS (all libraries)                          28

1981 GETS were required to define 28 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:RMOXPFN/OBJ=OBJ$:RMOXPFN MSRC$:RMOXPFN/UPDATE=(ENH$:RMOXPFN)+EXECML$/LIB+LIB$:RMS/LIB
```

0320 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0321 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY